LIMDEP

VERSION 11

Reference Guide

by

William H. Greene
Econometric Software, Inc.

Econometric Software, Inc.
15 Gloria Place
Plainview, NY 11803
USA

Tel:	+1 516-938-5254
Fax:	+1 516-938-2441
Email:	sales@limdep.com
Websites:	www.limdep.com and www.nlogit.com.

Econometric Software, Australia
6/16 Carr Street
Waverton NSW 2060
Australia

Tel:	+61 (0) 418-433-057
Email:	hgroup@hensher.com.au

Limited Warranty

Econometric Software warrants that the software product will perform substantially in accordance with the documentation for a period of ninety (90) days from the date of the original purchase. To make a warranty claim, you must notify Econometric Software in writing within ninety (90) days from the date of the original purchase and return the defective software to Econometric Software. If the software does not perform substantially in accordance with the documentation, the entire liability and your exclusive remedy shall be limited to, at Econometric Software's option, the replacement of the software product or refund of the license fee paid to Econometric Software for the software product. Proof of purchase from an authorized source is required. This limited warranty is void if failure of the software product has resulted from accident, abuse, or misapplication. Some states and jurisdictions do not allow limitations on the duration of an implied warranty, so the above limitation may not apply to you. To the extent permissible, any implied warranties on the software product are limited to ninety (90) days.

Econometric Software does not warrant the performance or results you may obtain by using the software product. To the maximum extent permitted by applicable law, Econometric Software disclaims all other warranties and conditions, either expressed or implied, including, but not limited to, implied warranties of merchantability, fitness for a particular purpose, title, and non-infringement with respect to the software product. This limited warranty gives you specific legal rights. You may have others, which vary from state to state and jurisdiction to jurisdiction.

Limitation of Liability

Under no circumstances will Econometric Software be liable to you or any other person for any indirect, special, incidental, or consequential damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, computer failure or malfunction, loss of business information, or any other pecuniary loss) arising out of the use or inability to use the software product, even if Econometric Software has been advised of the possibility of such damages. In any case, Econometric Software's entire liability under any provision of this agreement shall not exceed the amount paid to Econometric Software for the software product. Some states or jurisdictions do not allow the exclusion or limitation of liability for incidental or consequential damages, so the above limitation may not apply to you.

Preface

LIMDEP is a flexible computer package for estimating models for cross section and panel data. Its capabilities range from basic linear regression and descriptive statistics to many advanced techniques such as parametric duration models, Poisson regressions with right censoring, nonlinear regressions estimated by instrumental variables and the generalized method of moments (GMM) and nonlinear panel data models with random parameters and fixed effects. *LIMDEP*'s menu of options is as wide as that of any other general purpose program available. *LIMDEP* has provided many recent innovations in econometrics, including cutting edge techniques in panel data analysis, frontier and efficiency estimation and discrete choice modeling. The package also provides a programming language to allow the user to specify, estimate and analyze models that are not contained in the built in menus of model forms.

This program has developed since 1980, initially to provide an easy to use tobit estimator – hence the name, '***LIM***ited***DEP***endent variable models.' It has spun off a major suite of programs for the estimation of discrete choice models. This program, *NLOGIT*, builds on the ***N***ested ***LOGIT*** model. *NLOGIT* has now grown to a self standing superset of *LIMDEP*.

Version 11 of *LIMDEP* continues our efforts to produce major upgrades to the program while maintaining full compatibility with earlier versions. Version 11 features numerous new estimation programs and further refinements of the interface, a long list of enhancements to the user interface, additions to the **MATRIX**, **CALC** and **CREATE** programming elements, other data manipulation commands including some new handling of character data, and improvements to the internal workings of the mathematical parts of the program.

This release coincides with the release of *NLOGIT* Version 6, a superset of *LIMDEP* Version 11 which extends the standard discrete choice (multinomial logit) model contained in *LIMDEP* to many modifications and alternative specifications.

To the best of our knowledge, the code of this program is correct as described. However, no warranty is expressed or implied. Users assume responsibility for the selection of this program to achieve their desired results and for the results obtained.

Econometric Software, Inc.
Plainview, NY 11803
2016

Table of Contents

Table of Contents vi

What's New in Version 11? 1
WN1 New Models and Model Extensions 3
WN1.1 Stochastic Frontier Models for Cross Sections 3
WN1.2 Sample Selection Correction for Stochastic Frontier Model 4
WN1.3 Panel Data Stochastic Frontier Models 4
WN1.4 Partially Inefficient Latent Class Stochastic Frontier Model 8
WN1.5 Data Envelopment Analysis 9
WN1.6 Fractional Regression for Cross Section and Panel Data 10
WN1.7 Models with Endogenous Treatment Effects 10
WN1.8 Inverse Hyperbolic Sine Model 12
WN1.9 Loglinear Models 13
WN2 MAXIMIZE/MINIMIZE and Nonlinear Least Squares 13
WN2.1 Panel Data in MAXIMIZE 13
WN3 New and Expanded Data Descriptive Devices 14
WN3.1 Means 14
WN3.2 Correlations 14
WN3.3 Tables 14
WN3.4 Quantiles 15
WN3.5 Frequencies 16
WN4 Extensions of Graphical Descriptive Devices 16
WN4.1 Histograms 17
WN4.2 Box Plots 19
WN4.3 Kernel Density Estimator 19
WN4.4 Bubble Plot 20
WN5 PARTIALS and SIMULATE 20
WN5.1 Graphical Devices 20
WN5.2 Transition Matrices for Partial Effects of Categorical Variables 22
WN5.3 Krinsky and Robb Method 24
WN5.4 Customized Models and Extensions 24
WN6 Analysis Tools and Capabilities 25
WN7 Programming with EXECUTE and PROCEDURES 25
WN8 Importing and Using Character Variables 26
WN9 Missing Data 28

R1: Introduction to *LIMDEP* Version 11 30
R1.1 The *LIMDEP* Program 30
R1.2 Econometric Techniques 32
R1.3 Summary of What's New in Version 11 34
R1.4 Documentation 35

R2: Basics of Operation 36
R2.1 Introduction to the *LIMDEP* Desktop 36

R2.1.1 *LIMDEP* Desktop Menus........ 36
R2.1.2 The *LIMDEP* Toolbar and Command Bar........ 37
R2.1.3 Components of a *LIMDEP* Session........ 38
R2.2 *LIMDEP* File Types........ 38
R2.3 Beginning the *LIMDEP* Session........ 40
R2.3.1 Opening a Project File........ 40
R2.3.2 Opening an Editing Window........ 40
R2.4 Using the Editing Window........ 42
R2.4.1 Using the Insert Menu in the Editing Window........ 43
R2.4.2 Executing Commands from the Editing Window........ 44
R2.4.3 The Editing Window Right Mouse Button Menu........ 45
R2.5 A Short Tutorial........ 46
R2.6 Commands........ 53
R2.6.1 Syntax........ 53
R2.6.2 Naming Conventions and Reserved Names........ 54
R2.7 Input Files - Entering Commands from a File........ 55
R2.8 Work Areas and Projects........ 57
R2.8.1 Work Areas........ 57
R2.8.2 The Project Window........ 59
R2.9 Restarting During a Session........ 64
R2.10 Program Output and the Output Window........ 65
R2.10.1 Opening an Output File........ 66
R2.10.2 Editing Your Output........ 66
R2.10.3 Printing........ 66
R2.11 Help........ 67
R2.12 Summary of Commands........ 69
R2.13 Summary of the *LIMDEP* Desktop........ 72
R2.13.1 The *LIMDEP* Windows........ 73
R2.13.2 The Main Menus........ 74
R2.13.3 The *LIMDEP* Toolbar........ 80
R2.13.4 The Command Bar........ 80
R2.13.5 Commands and Menu Items........ 81

R3: Importing and Reading Data Files........ 82
R3.1 Importing and Reading Data........ 82
R3.2 Import a Standard Formatted ASCII File........ 82
R3.2.1 Observation Labels and Variable Names in the Data File........ 85
R3.2.2 Alphabetic Tags for Observations........ 86
R3.2.3 Data Files that Contain Only Numeric Data........ 87
R3.2.4 Observation Labels without Variable Names in the Data File........ 88
R3.2.5 Reading a Spreadsheet File from *Excel*........ 88
R3.2.6 Missing Values in Data Files........ 91
R3.2.7 Missing Values in the Comma Delimited (CSV) Files........ 91
R3.2.8 Changing the Missing Value Code........ 92
R3.2.9 Data Files that Are Not Formatted for IMPORT........ 92
R3.3 The Data Editor........ 93
R3.4 The Data Area........ 98

R3.4.1 Temporary Expansion of the Data Area 98
R3.4.2 Permanently Setting the Number of Cells in the Data Area 99
R3.4.3 Setting the Number of Rows in the Data Area........ 100
R3.5 The READ Command for Nonstandard Data Files 101
R3.5.1 ASCII Numeric Data Files........ 101
R3.5.2 Variable Names Not Provided in the Data File........ 102
R3.5.3 Variable Names in the Data File........ 102
R3.5.4 Observation Labels 103
R3.5.5 Transposed Data Files – Reading by Variables 104
R3.5.6 Binary Files and Files from Other Programs 105
R3.5.7 Formatted ASCII Files........ 107
R3.5.8 Recoding Character Data 109
R3.6 Using the Text Editor as a Data File........ 110
R3.6.1 Use the Text Editor to Avoid Creating a Data File........ 111
R3.6.2 Exporting from *Excel* to the Text Editor........ 112
R3.7 Documenting the Contents of a Data/Project File 116
R3.8 Listing Data in Your Output Window........ 117
R3.9 Exporting and Writing Data Files 119
R3.9.1 How to EXPORT a CSV File 119
R3.9.2 How to WRITE a Data File 121
R3.10 Adding Observations – The APPEND Command 122

R4: Data Transformations 124
R4.1 Data Transformations........ 124
R4.2 The CREATE Command 124
R4.2.1 Algebraic Transformations 126
R4.2.2 Conditional Transformations 130
R4.3 CREATE Functions 133
R4.3.1 Common Algebraic Functions 133
R4.3.2 Univariate Normal and Related Distributions........ 134
R4.3.3 Logistic and Discrete Distributions 134
R4.3.4 Trends and Seasonal Dummy Variables 134
R4.3.5 Ranks of Observations 135
R4.3.6 Box-Cox Function and its Derivatives........ 135
R4.3.7 Bivariate and Multivariate Normal Probabilities 136
R4.3.8 Leads and Lags 137
R4.3.9 Matrix Functions........ 138
R4.3.10 Moving a Matrix 139
R4.3.11 Means, Deviations, Standardized Variables 139
R4.3.12 Regression Residuals and Fitted Values 140
R4.3.13 Moments for a Set of Variables – the Xmt Function 140
R4.3.14 Multiple of a Set of Variables – the Scl Function........ 141
R4.3.15 Expanding a Categorical Variable into a Set of Dummy Variables........ 142
R4.3.16 Stacking Data to Create Data Matrices........ 145
R4.3.17 Group Functions for Panel Data 145
R4.3.18 Functions Based on Groups Using Observation Tags 148
R4.3.19 Aggregating Grouped or Stratified Data........ 151

R4.4 Random Number Generators 152
R4.4.1 Setting the Seed for the Random Number Generator 153
R4.4.2 Basic Random Number Generation 153
R4.4.3 Random Samples from Continuous Distributions 155
R4.4.4 Random Samples from Discrete Distributions........ 156
R4.4.5 Sampling from the Multivariate Normal Distribution 157
R4.5 Compound Names for Variables........ 159
R4.6 Changing Particular Observations of a Variable........ 161
R4.7 Recoding Variables – The RECODE Command 161
R4.8 Sorting Variables – The SORT Command 162
R4.9 The DELETE and RENAME Commands 164
Appendix R4A Numerical Methods 165
R4A.1 Computing Bivariate Normal Probabilities........ 165
R4A.2 Computing Multivariate Normal Probabilities........ 165
R4A.3 Uniform Random Number Generation........ 166
R4A.4 Standard Normal Random Number Generation 167
R4A.5 Random Number Generation from Other Distributions........ 167
R4A.6 Sampling from the Truncated Normal Distribution 168
R4A.7 Random Sampling from the Multivariate Normal Distribution 168
R4A.8 Sample Variances........ 168

R5: Panel Data and Data for Discrete Choice Models 169
R5.1 Estimation Using Panel Data 169
R5.2 Programs that Use Panel Data........ 169
R5.3 Panel Data Arrangement 170
R5.3.1 Group Indicators and Within Group Observation Numbers 171
R5.3.2 Group Size Variables for Panel Data 173
R5.3.3 Permanent Global Setting for Panel Data 174
R5.4 Merging Invariant Variables into a Panel Data Set........ 176
R5.4.1 Using an ID Variable to Merge Data 177
R5.4.2 Using a Group Count Variable to Merge Data 178

R6: Variable Lists and Labellists 180
R6.1 Namelists and Labellists 180
R6.2 Lists of Variables in Model Commands 180
R6.3 Wildcard Characters in Variable Lists 180
R6.4 Defining Namelists 181
R6.4.1 Combining and Shortening Namelists 181
R6.4.2 Deleting Namelists........ 182
R6.4.3 Editing Namelists........ 183
R6.5 Using Namelists........ 184
R6.5.1 Using Namelists in Commands........ 184
R6.5.2 Interaction Terms in Models and Partials 185
R6.5.3 Using Namelists in Matrix Algebra 186
R6.5.4 Using Namelists to Display Model Results 186
R6.5.5 Using Namelists in CREATE 187
R6.5.6 Using NAMELIST to Create a Data Matrix 188

R6.5.7 Indexing Variables in Namelists 188
R6.6 Labellists 189

R7: The Current Sample and Missing Data 191
R7.1 The Current Sample 191
R7.2 Cross Section Data 193
R7.2.1 Defining the Current Sample with the SAMPLE Command 193
R7.2.2 Removing and Adding Observations with REJECT/INCLUDE 193
R7.2.3 Interaction of REJECT/INCLUDE and SAMPLE 195
R7.2.4 Using Observation Tags to Set the Current Sample 196
R7.3 Time Series Data 196
R7.4 Using the DRAW Command to Obtain Random Samples 198
R7.4.1 Random Sampling from a Cross Section 198
R7.4.2 Random Sampling from a Panel Data Set 200
R7.4.3 Simulating a Random Sample with Panel Data 200
R7.5 Missing Data 201
R7.5.1 Reading Missing Data 201
R7.5.2 Missing Data in Transformations 202
R7.5.3 Missing Data in Scalar and Matrix Algebra 202
R7.5.4 Missing Data in Estimation Routines 203
R7.5.5 Automatically Bypassing Missing Data – The SKIP Command 204
R7.5.6 Nonlinear Optimization Programs and Using SKIP Generally 206
R7.6 The Sample Used for the Most Recent Model 206

R8: Commands for Estimating Models 207
R8.1 Model Specifications of Variables and Weights 207
R8.2 Model Commands 207
R8.3 Interaction Terms and Nonlinear Functions of Variables 209
R8.3.1 Interaction Terms and Logs of Variables in Commands 210
R8.3.2 Interaction Terms and Nonlinear Terms in Namelists 213
R8.3.3 Managing Constructed Variables in the Data Set 213
R8.4 Categorical Variables in Model Commands 214
R8.5 Lags and Partial Differences in Model Commands 216
R8.6 Command Builders 218
R8.7 Conditional Model Commands 222
R8.7.1 Estimation Conditioned on a Scalar Test Value 222
R8.7.2 Setting the Sample Temporarily for a Model 223
R8.7.3 Looping over Strata for a Model Command 223
R8.8 Using Weights in Estimation 225

R9: Output 226
R9.1 Standard Output from Estimation Programs 226
R9.1.1 Changing the Confidence Level for the Confidence Intervals 228
R9.1.2 Information Criteria for Maximum Likelihood Estimators 229
R9.1.3 Timing Model Estimation 230
R9.2 Initial Model Results 231
R9.2.1 Displaying Initial Least Squares Estimates 231

R9.2.2 Intermediate Model Estimates 232
R9.3 Using DISPLAY to View Estimation Results 233
R9.4 Covariance Matrices, Predictions and Hypothesis Tests 234
R9.4.1 Displaying Covariance Matrices 234
R9.4.2 Listing and Saving Model Predictions and Residuals 235
R9.4.3 Listing Basic Partial Effects 237
R9.4.4 Hypothesis Tests and Restrictions 238
R9.4.5 Graphical Results 240
R9.5 Suppressing Results 240
R9.5.1 Suppressing Estimation Results with Quiet 241
R9.5.2 Suppressing All Results with SILENT 242
R9.6 The Review Window – Tables of Model Results 244
R9.7 Output Files 246
R9.7.1 Transporting Output Results to Word Processors 247
R9.7.2 Exporting Statistical Results from *LIMDEP* 248
R9.7.3 The Last Model Output 250

R10: Robust Covariance Matrices and Clustering 251
R10.1 Robust Covariance Matrix for Pooled Models 251
R10.2 Using Clustering for Robust Covariance Matrices 253
R10.2.1 Models for Which the Clustering Estimator is Supported 254
R10.2.2 An Example of the Clustering Estimator 255
R10.2.3 Technical Details on the Clustering Estimator 256
R10.3 Stratified and Grouped Data 258

R11: Partial Effects 261
R11.1 Partial Effects for Estimated Models 261
R11.2 Command vs. Model Specification 262
R11.3 Partial Effects Issues 268
R11.4 The PARTIAL EFFECTS Command 273
R11.4.1 Last Model Used for Partial Effects 276
R11.4.2 Sample Used for PARTIAL EFFECTS 277
R11.4.3 Types of Variables in Partial Effects 277
R11.4.4 Types of Partial Effects 278
R11.4.5 Scenarios in the PARTIAL EFFECTS Command 280
R11.4.6 Plotting Partial Effects 282
R11.4.7 Sample Partitioning: The '@' Specification 284
R11.4.8 Fixing Variables for the Entire Analysis 286
R11.4.9 Saving Individual Partial Effects 286
R11.4.10 Partial Effects for Categorical Variables 287
R11.4.11 Computing Partial Effects at Sample Means 289
R11.4.12 Weighted Observations 291
R11.4.13 Robust Covariance Matrices 291
R11.4.14 Changing the Model Analyzed by PARTIAL EFFECTS 291
R11.4.15 Technical Details: The Delta Method and Krinsky and Robb 294
R11.5 Partial Effects Estimated with Models 296
R11.5.1 Partial Effects for Single Index Models 297

R11.5.2 Partial Effects for Dummy Rhs Variables 299
R11.5.3 Standard Errors and Confidence Intervals 300
R11.5.4 Significance Tests for Partial Effects 300
R11.5.5 Partial Effects in Compound Models 302
R11.5.6 Partial Effects in a Two Equation Model 303
R11.5.7 Partial Effects in a Model with Direct and Indirect Effects 306

R12: Model Predictions, Residuals, Simulations and Decompositions 307
R12.1 Introduction 307
R12.2 Creating and Displaying Predictions and Residuals 307
R12.3 The Last Model 311
R12.4 Using SIMULATE with the Last Model 312
R12.4.1 The Sample Used in the Simulation 313
R12.4.2 Scenarios in Simulations 313
R12.4.3 Difference in Differences Analysis 315
R12.4.4 Defining the Model for SIMULATE 316
R12.4.5 Keeping the Simulation Results as a Variable 319
R12.4.6 The Distribution of the Simulated Values 319
R12.5 Oaxaca-Blinder Decompositions 320

R13: Testing Hypotheses and Imposing Restrictions 325
R13.1 Introduction 325
R13.2 F Test of Linear Restrictions in Linear Models 325
R13.3 Testing Linear Restrictions Using the Wald Statistic 328
R13.4 Likelihood Ratio Tests 332
R13.4.1 Fixed Value Restriction in a Poisson Model 333
R13.4.2 Imposing and Testing Restrictions 333
R13.4.3 Homogeneity of Models in a Stratified Data Set 334
R13.4.4 Testing for Equal Coefficient Vectors 335
R13.4.5 Two Part Models: Cragg’s Model for a Censored Dependent Variable 336
R13.4.6 Likelihood Ratio Tests for Discrete Choice Models 337
R13.4.7 Likelihood Ratio Tests for Nonlinear Models 338
R13.5 Lagrange Multiplier Tests 339
R13.5.1 LM Tests Based on the Model Specification 339
R13.5.2 LM Test of Homoscedasticity in a Probit Model 342
R13.5.3 LM Tests for the Linear Regression Model 344
R13.5.4 Programming Lagrange Multiplier Tests 345
R13.6 Estimation Subject to Restrictions 347
R13.6.1 Fixed Value and Equality Restrictions 348
R13.6.2 General Linear Restrictions 352
R13.6.3 Imposing Linear Constraints on Maximum Likelihood Estimators 353
R13.6.4 Restricted Linear Regression with Multicollinearity 355

R14: Functions of Parameters 358
R14.1 Introduction 358
R14.2 Covariance Matrices for Nonlinear Functions 358
R14.2.1 The Delta Method 358

R14.2.2 The Method of Krinsky and Robb 359
R14.3 The Wald Statistic 360
R14.4 The WALD Command 360
R14.4.1 Components of the WALD Command 361
R14.4.2 Results of the WALD Command 362
R14.4.3 Recursive Functions 363
R14.4.4 Application Based on the Last Model 364
R14.4.5 The Number of Parameters 365
R14.4.6 Interdependent Sets of Functions 365
R14.4.7 Extracting Parts of a Model 367
R14.4.8 Application to a Function of the Parameters 367
R14.4.9 Application to a Complex Nonlinear Function 371

R15: Retrievable Results 372
R15.1 Introduction 372
R15.2 Retrievable Results 374

R16: Using Matrix Algebra 379
R16.1 Introduction 379
R16.2 Entering MATRIX Commands 389
R16.2.1 The Matrix Calculator 389
R16.2.2 MATRIX Commands 391
R16.2.3 Conditional Commands 392
R16.3 Matrix Output 393
R16.3.1 Matrix Results 394
R16.3.2 Unformatted Output 395
R16.3.3 Technical Output 396
R16.3.4 Exporting Matrix Results from *LIMDEP* 396
R16.3.5 Matrix Statistical Output 399
R16.3.6 Descriptive Statistics for the Elements in a Matrix 401
R16.3.7 Plotting Matrices 401
R16.4 Matrix Work Areas 401
R16.4.1 Rebuilding the Matrix Work Area 402
R16.4.2 Naming and Notational Conventions 402
R16.4.3 Matrix Dimensions 404
R16.4.4 Placing Matrix Results in Scalars 404
R16.4.5 Compound Names for Matrices, Variables and Scalars 405
R16.5 Reading Matrices 407
R16.5.1 Importing a Matrix as a Data File 407
R16.5.2 Importing a Matrix as a Block of Cells from *Excel* 408
R16.6 Matrix Expressions 409
R16.6.1 Scalar Multiplication of a Result – Using CALCULATE 412
R16.6.2 Adding the Same Scalar to Every Element of a Matrix 413
R16.6.3 Raising a Matrix to a Power 414
R16.6.4 Entering, Moving, and Rearranging Matrices 415
R16.7 Using MATRIX Commands with Data 422
R16.7.1 Data Matrices 422

R16.7.2 Computations Involving Data Matrices 424
R16.8 Functions for Manipulating Matrices 425
R16.8.1 Functions of One Matrix 426
R16.8.2 Functions of Two or More Matrices 429
R16.9 Sums of Observations 430
R16.10 Matrix Commands that Transform the Data 438
R16.10.1 Linear Transformations of Variables 438
R16.10.2 Moving a Matrix into the Data Area 439
R16.11 MATRIX Commands for Panel Data 440
R16.11.1 MATRIX Functions for Panel Data 440
R16.11.2 GMM Weighting Matrix for Panel Data 442
R16.11.3 Gsum and Gmmw Functions with Weights for Some or All Variables 442
R16.11.4 Matrix Forms for Computing Moments for Panel Data 443

R17: Using the Calculator 444
R17.1 Introduction 444
R17.2 Command Input in CALCULATE 445
R17.3 Results from CALCULATE 447
R17.4 Forms of CALCULATE Commands – Conditional Commands 449
R17.4.1 Reserved Names 449
R17.4.2 Work Space for the Calculator 450
R17.4.3 Compound Names for Scalars 451
R17.5 Scalar Expressions 452
R17.6 Calculator Functions 453
R17.6.1 Basic Algebraic and Trigonometric Functions 453
R17.6.2 Relational Functions 454
R17.6.3 Critical Points from the Normal Family of Distributions 454
R17.6.4 Probabilities and Densities for Continuous Distributions 454
R17.6.5 Moments of the Left Truncated Normal Distribution 455
R17.6.6 Probabilities and Densities for the Bivariate Normal Distribution 455
R17.6.7 Probabilities and Densities for the Multivariate Normal Distribution 455
R17.6.8 Probabilities for Noncentral Distributions 456
R17.6.9 Probabilities for Discrete Distributions 456
R17.6.10 Gamma Function and Gamma Distribution 456
R17.6.11 The Incomplete Gamma Function 457
R17.6.12 Random Numbers 457
R17.6.13 Matrix Dimensions and Functions 457
R17.6.14 Sample Statistics and Regression Results 458
R17.7 Fit Measures for a Binary Choice Model 461
R17.8 Hypothesis Tests 463
R17.9 Calculating Correlation Coefficients 466
R17.10 Augmented Dickey Fuller Test 471
R17.11 Plotting Discrete Distributions 472

R18: Two Step Estimators 473
R18.1 Covariance Matrices for Two Step Estimation 473
R18.2 Two Step Estimation for an Endogenous Discrete Variable 474

R18.3 Two Step Estimation for an Endogenous Regression Variable 476
R18.4 Programming a Two Step Estimator 477
R18.5 Theory for Two Step Estimators 480

R19: Programming with Procedures 484
R19.1 Introduction 484
R19.2 The Text Editor 484
R19.2.1 Placing Commands in the Editor 484
R19.2.2 Executing the Commands in the Editor 485
R19.2.3 Executing Model Commands Quietly 487
R19.2.4 Using Text Files with the Editor 488
R19.3 Estimation Programs and Postprocessing 490
R19.4 Procedures 492
R19.5 Defining and Executing Procedures 494
R19.5.1 The Procedure Library 495
R19.5.2 Executing a Procedure 496
R19.5.3 Repeated Execution of a Procedure 498
R19.5.4 Execution with a Scalar Parameter 498
R19.5.5 Query for a Parameter to Use in the Procedure 499
R19.5.6 Conditional Execution 500
R19.5.7 Defining Exit (Convergence) Criteria 502
R19.5.8 Parameters and Character Strings in Procedures 504
R19.5.9 Local Variables in Procedures 506
R19.6 Looping with the EXECUTE Command 508
R19.7 Looping Over an Indexed Set 508
R19.7.1 Looping Over a Set of Variables 508
R19.7.2 Looping over Observation Tags 509
R19.8 Flow Control within Procedures 510
R19.8.1 Logical Expressions 510
R19.8.2 Loops within Procedures 511
R19.9 Looping with DO Statements 513
R19.10 Escaping from an Infinitely Looping Procedure 515
R19.11 Editing Procedures and Creating New Procedures 516

R20: Multiple Imputation 518
R20.1 Introduction to Multiple Imputation 518
R20.2 Methodology 519
R20.3 How It’s Done – Overview 520
R20.4 The Imputation Step 522
R20.5 The Estimation Step 523
R20.6 The Aggregation Step and Post Estimation Analysis 527
R20.7 Using Multiple Imputation in Your Own Model 528
R20.8 Imputation Methods 528
R20.9 Usage Notes 530
R20.9.1 Questions on Usage 530
R20.9.2 Implementation Notes 532

R21: Bootstrapping and Other Sampling Experiments 533
R21.1 Introduction 533
R21.2 Bootstrapping Cross Sections and Panel Data 533
R21.3 Jackknife Estimation 539
R21.4 Random Sampling from the Current Sample – DRAW 542
R21.5 Random Sampling from Panel Data Sets 543
R21.6 Random Number Generators 544
R21.6.1 Setting the Seed for the Random Number Generator 544
R21.6.2 Using CREATE to Generate Random Samples 544
R21.6.3 Sampling from the Multivariate Normal Distribution 546
R21.6.4 Using CALC to Generate Random Draws 548
R21.6.5 Using MATRIX to Draw Random Matrices 549
R21.6.6 Simulating Random Effects in a Panel 549
R21.6.7 Simulating an Unbalanced Panel Data Set 550
R21.7 Plotting Distributions 551
R21.7.1 CALC Functions that Show Discrete Distributions 551
R21.7.2 Plotting a Density 552
R21.7.3 Drawing a Distribution by Plotting a Histogram 553
R21.7.4 Sampling Experiments 554
R21.7.5 The Law of Large Numbers and the Central Limit Theorem 555
R21.8 Urn Experiments 558

R22: Models for Panel Data 562
R22.1 Introduction 562
R22.2 Panel Data Models 562
R22.3 Data Arrangement and Setup 563
R22.3.1 Data Arrangement 564
R22.3.2 Reordering Balanced Panels 564
R22.3.3 CREATE Commands for Panel Data 565
R22.4 General Model Forms for Panel Data 566
R22.5 Model Commands 569
R22.5.1 Specifying the Panel 569
R22.5.2 Missing Data 570
R22.5.3 Model Type Specifications, Output and Saved Matrices 570

R23: Fixed and Random Effects Models for Panel Data 581
R23.1 Introduction 581
R23.2 Fixed Effects Models 581
R23.2.1 Least Squares in the Linear Regression Model 581
R23.2.2 Maximum Likelihood Estimation 585
R23.2.3 How it's Done 588
R23.3 Random Effects Models 593
R23.3.1 Quadrature Based Estimation – The Butler and Moffitt Method 595

R24: Random Parameter Models 600
R24.1 Random Parameters Models 600
R24.2 Mathematical Formulation of the RP Model 601

R24.3 Commands for Random Parameters Models 602
R24.4 The Parameter Vector and Starting Values 608
R24.5 Individual Specific ‘Estimates’ 609
R24.6 Application 612
R24.7 Technical Details on Estimation of RP Models by Simulation 616
R24.8 Multilevel and Multiple Effects RP Models 631
R24.8.1 Command 632
R24.8.2 Application 632
R24.8.3 Technical Details 636

R25: Latent Class Models 637
R25.1 Latent Class Models 637
R25.2 Commands for Latent Class Modeling 638
R25.3 Modeling Frameworks for Latent Class Analysis 639
R25.4 Output and Saved Results 639
R25.5 Extending the Class Probability Model 642
R25.6 Testing for the Latent Class Model 644
R25.7 Individual Specific Estimates 645
R25.7.1 Individual Specific Posterior Class Probabilities 646
R25.7.2 Individual Specific Parameters 647
R25.8 Application 648
R25.9 Technical Details on Estimating Latent Class Models 651

R26: Numerical Optimization 653
R26.1 Numerical Optimization 653
R26.2 Technical Display During Optimization 653
R26.3 Technical Output During Iterations 654
R26.4 Exit from Iterations and Warning Messages 657
R26.4.1 Normal Exit from Iterations 657
R26.4.2 Maximum Iterations 657
R26.4.3 Unable to Find Function Optimum 658
R26.4.4 Too Few Iterations 660
R26.4.5 General Failure of Indeterminate Cause 660
R26.4.6 Interrupting the Iterations 661
R26.4.7 Warnings During the Iterations 662
R26.5 Exit Codes 662
R26.6 Iteration Controls 663
R26.6.1 Maximum Iterations 663
R26.6.2 Algorithms 663
R26.6.3 Convergence Rules 664
R26.7 Quadrature 665
R26.8 Multivariate Normal Probabilities 666
R26.8.1 Model Based on the Multivariate Normal Distribution 667
R26.8.2 Tools that Calculate Multivariate Normal Probabilities 667
R26.9 Default Values of Program Parameters 669
R26.10 Starting Values 670
R26.11 Hints for Iterative Estimation 671

Appendix R26A Technical Details on Optimization 672

R27: Summary for *LIMDEP Reference Guide* 675
R27.1 Introduction 675
R27.2 Essential Program Functions 675
R27.2.1 Startup 675
R27.2.2 Operation 675
R27.3 Reading a Data Set 676
R27.4 Transforming Data 676
R27.5 Setting the Sample 677
R27.6 Multiple Imputation 678
R27.7 Econometric Model Estimation 678
R27.7.1 Variable Specifications in Model Commands 679
R27.7.2 Controlling Output from Model Commands 680
R27.7.3 Robust Asymptotic Covariance Matrices 680
R27.7.4 Optimization Controls for Nonlinear Optimization 681
R27.7.5 Setup for Simulation Based Estimators 681
R27.7.6 Execution of Procedures for Model Estimation 681
R27.7.7 Predictions and Residuals 681
R27.7.8 Model Setup for Certain Models 681
R27.7.9 Setup for Panel Data Models 682
R27.8 Post Estimation 683
R27.8.1 Hypothesis Tests and Restrictions 683
R27.8.2 Partial Effects 683
R27.8.3 Oaxaca Decompositions 683
R27.9 The Command Builders 684
R27.10 Econometric Data Structures and Modeling Tools 687
R27.10.1 Cross Section Data 687
R27.10.2 Panel Data 687
R27.10.3 Fixed Effects Models 690
R27.10.4 Random Effects and Multilevel Random Effects Models 691
R27.10.5 The Random Parameters Model 693
R27.10.6 Observations About GLIM and GEE Estimation 694
R27.10.7 Latent Class Models 696
R27.10.8 Time Series Data 697
R27.11 Econometric Model Estimation Templates 697

R28: Diagnostics and Error Messages 698
R28.1 Introduction 698
R28.2 Optimization 699
R28.3 Setup and Runtime Diagnostics 700
R28.4 Discrete Choice (CLOGIT) and NLOGIT 751

***LIMDEP* 11 References 759**

***LIMDEP* 11 *Reference Guide* Index 777**

What's New in Version 11?

LIMDEP Version 11 contains estimators for several new models, many extensions of the present models and additions to the kit of tools for data analysis. We continue to develop the **SIMULATE** and **PARTIAL EFFECTS** commands that should change the way you build models with your data. These capabilities apply to every model that you can estimate including those you design yourself with **MAXIMIZE** and to any function you choose to specify even if it is not a component of a model. There are also many extensions of existing models and commands as well as enhancements to basic tools such as panel data handling, data transformations, matrix algebra, import of data, and so on.

WN1 New Models and Model Extensions

- Stochastic Frontiers
- Count Data
- Inverse Hyperbolic Sine with Hurdle Effects
- Treatment Effects
- Panel Data Extensions for fractional responses
- Loglinear models

WN2 MAXIMIZE/MINIMIZE and Nonlinear Least Squares

- Panel Data
- Nonlinear Least Squares with Random Parameters

WN3 New and Expanded Data Descriptive Statistics

- Means and correlations
- Quantiles
- Frequencies
- Cronbach's Alpha and Scale Variables

WN4 Graphical Descriptive Devices

- Histograms
- Box Plots
- Kernel Density Estimator
- Plotting

WN5 PARTIALS and SIMULATE

- Graphical Devices
- Transition Tables for Categorical Variables
- Krinsky and Robb Method
- Expanded Model Set
- Customized Models and Extensions

WN6 Analysis Tools and Capabilities

- **CALCULATE**
- **MATRIX**
- **CREATE** for Data Transformations

WN7 Programming with EXECUTE and PROCEDURES

WN8 Importing and Using Character Variables

- Mixed Data Sets
- **CREATE** Manipulations of Character Variables
- Controlling the Sample
- Iterating over Strata
- Labels and Labellists

WN9 Missing Data

WN1 New Models and Model Extensions

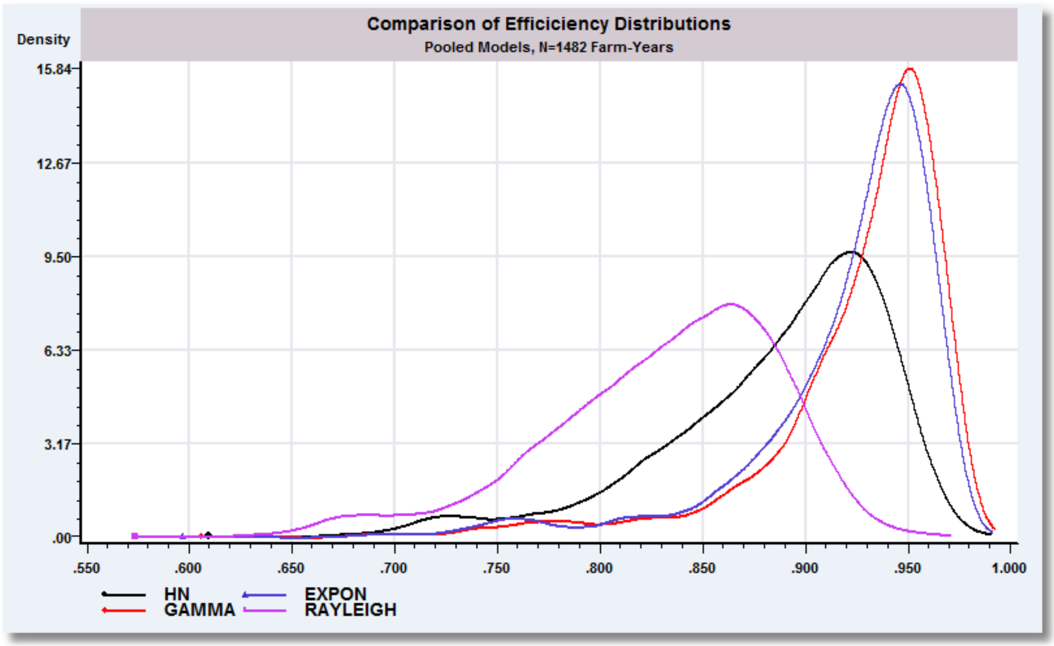
There are about 10 new models and model extensions in Version 11 of *LIMDEP*. We have also added new tools to **SIMULATE** and **PARTIAL EFFECTS** for extending the existing model specifications.

WN1.1 Stochastic Frontier Models for Cross Sections

There is a wide variety of parametric specifications for stochastic frontier models already provided in *LIMDEP*. We have added the normal-Rayleigh model (Hajargasht (2015)). This is a flexible one parameter functional form that allows for a greater range of shapes for the inefficiency distribution than the half normal or exponential. An advantage of the Rayleigh model is that the mode of the inefficiency distribution is away from zero, unlike the half normal and exponential models. The following example compares the estimated distribution of firm (farm) specific estimates based on these three models and a fourth, the normal-gamma model. The estimation commands compute the model then obtain the sample of estimated efficiency values. The distributions of these four samples are compared with a kernel density estimator.

```
NAMELIST    ; x = one,x1,x2,x3,x4 $
FRONTIER    ; Lhs = yit ; Rhs = x ; Techeff = hn $
FRONTIER    ; Lhs = yit ; Rhs = x ; Techeff = gamma    ; Model = Gamma $
FRONTIER    ; Lhs = yit ; Rhs = x ; Techeff = expon    ; Model = Exponential $
FRONTIER    ; Lhs = yit ; Rhs = x ; Techeff = rayleigh ; Model = Rayleigh $
KERNEL      ; Rhs = hn,gamma,expon,rayleigh
            ; Grid
            ; Title = Comparison of Efficiency Distributions
            ; Subtitle = Pooled Models, N = 1482 Farm-Years $
```

```
-------------------------------------------------------------
Kernel Density Estimator
Kernel Function     =  Epanechnikov
Results matrix      =        KERNEL
Observations        =          1482
Points plotted      =          1482
-------------------------------------------------------------
Variables Analyzed =            HN     GAMMA     EXPON RAYLEIGH
Bandwidth          =         .0116     .0099     .0100    .0119
Mean               =         .8854     .9253     .9203    .8307
Standard Deviation =         .0557     .0472     .0478    .0571
Skewness           =      -1.2674   -2.3479   -2.2712   -.8554
Kurtosis-3 (excess)=        1.6781    6.9397    6.4936    .7574
Chi2 normality test=       15.7262  128.0687  114.2594   5.7983
Minimum            =         .6209     .6152     .6067    .5851
Maximum            =         .9793     .9831     .9816    .9597
-------------------------------------------------------------
```



WN1.2 Sample Selection Correction for Stochastic Frontier Model

The sample selection corrected version of the normal-half normal stochastic frontier model (Greene (2010)) was introduced in Version 10. The proposed estimator is based on maximum simulated likelihood. Lai (2015) has developed a direct approach based on the full likelihood function (without simulation). The new FIML estimator is added to *LIMDEP* in Version 11. (The two estimators produce generally the same results, as might be expected.)

A panel data version of the model has been provided in Version 11. The panel data approach suggested here treats the selection as a permanent 'treatment.' Individuals are nonrandomly selected into the regime at the beginning of the observation period and remain in the treatment group for the full panel. The two estimators for the cross section specification, maximum simulated likelihood and FIML, are both provided for this model as well.

WN1.3 Panel Data Stochastic Frontier Models

The parametric stochastic frontier models, normal-half normal, normal-exponential, and so on, are extended to the several panel data specifications, including true random, true fixed effects, random parameters and latent class. The Rayleigh model described above has been included in this set of panel data models.

The true fixed effects model (Greene, 2005) has attracted considerable attention. The estimator in its most basic form, despite its appeal as a specification, appears to be affected by the well known 'incidental parameters problem.' This is a built in inconsistency of the maximum likelihood estimator that is inversely related to the number of periods in the panel. The incidental parameters problem shows up in different ways in particular models. In the stochastic frontier model, it appears to affect the estimators of the crucial variance parameters that are needed for computing technical efficiency. The 'true fixed effects' model was provided in *LIMDEP* 10 (notwithstanding the incidental parameters issue). Several researchers have since pursued different approaches to estimation of the model while avoiding the IP issue. Chen et al. (2014) have devised a full likelihood function for the within transformed model. While cumbersome, their approach does offer a solution. Wikstrom (2016) suggests an alternative based on the method of moments. Finally, Belotti and Ilardi (2014) have developed two further alternative approaches, one based on a simulation approach to a first differences estimator and one based on pairwise differences of the observations. The Wikstrom and Belotti approaches have been built into Version 11 of *LIMDEP*. Here is an example of Belotti and Ilardi's PDE estimation.

```
-----------------------------------------------------------------------------
Pairwise Difference Estimator for SF-FEM
Dependent variable                    YIT
Log likelihood function      2757.33321
Estimation based on N =   1482, K =   6
Inf.Cr.AIC  =  -5502.7 AIC/N =   -3.713
---------------------------------------
Parameters of Stochastic  Prod Frontier
Component  Sigma    Hetero. Avrg. Value
v(i,t)    .081261 1.000000      .081261
u(i,t)    .003414 1.000000      .003414
Lambda  = sigmau/sigmav      =  .042019
Sigma^2 = sigmau^2+sigmav^2  =  .006615
Sigma                        =  .081333
Gamma   = sigmau^2/sigma^2   =  .001762
Estd FE Alpha(i) saved as var. ALPHA_I_
---------------------------------------
Panel Data Used for PDE Estimation ....
Number of Firms =  247 AgvT(i)  =   6.0
Group Size:  Minimum =  6, Maximum =  6
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Production/Cost/Distance Function Parameters........................
      X1|     .66198***      .02484    26.65  .0000       .61330    .71066
      X2|     .03735*        .01978     1.89  .0590     -.00142    .07612
      X3|     .03041         .02323     1.31  .1905     -.01512    .07595
      X4|     .38251***      .01186    32.26  .0000       .35927    .40576
--------+--------------------------------------------------------------------
        |Standard Deviation of Normal Disturbance............................
Sigma(v)|     .08126***      .00194    41.97  .0000       .07747    .08506
        |Standard Deviation of Underlying Inefficiency Distribution..........
Sigma(u)|     .00341         .07429      .05  .9633     -.14220    .14903
--------+--------------------------------------------------------------------
***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The true random effects (TRE) stochastic frontier model was proposed in Greene (2004a) and incorporated in Version 10 of *LIMDEP*. The TRE model has a time invariant random effect embedded in a stochastic frontier model with both time varying 'noise' and inefficiency terms. The generalized TRE model was suggested in Colombi (2010) and Kumbhakar, Lein and Hardaker, (2014). The generalized model adds a time invariant inefficiency term to the TRE model, producing the specification

$$y_{it} = \alpha_i - w_i + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it} - u_{it}.$$

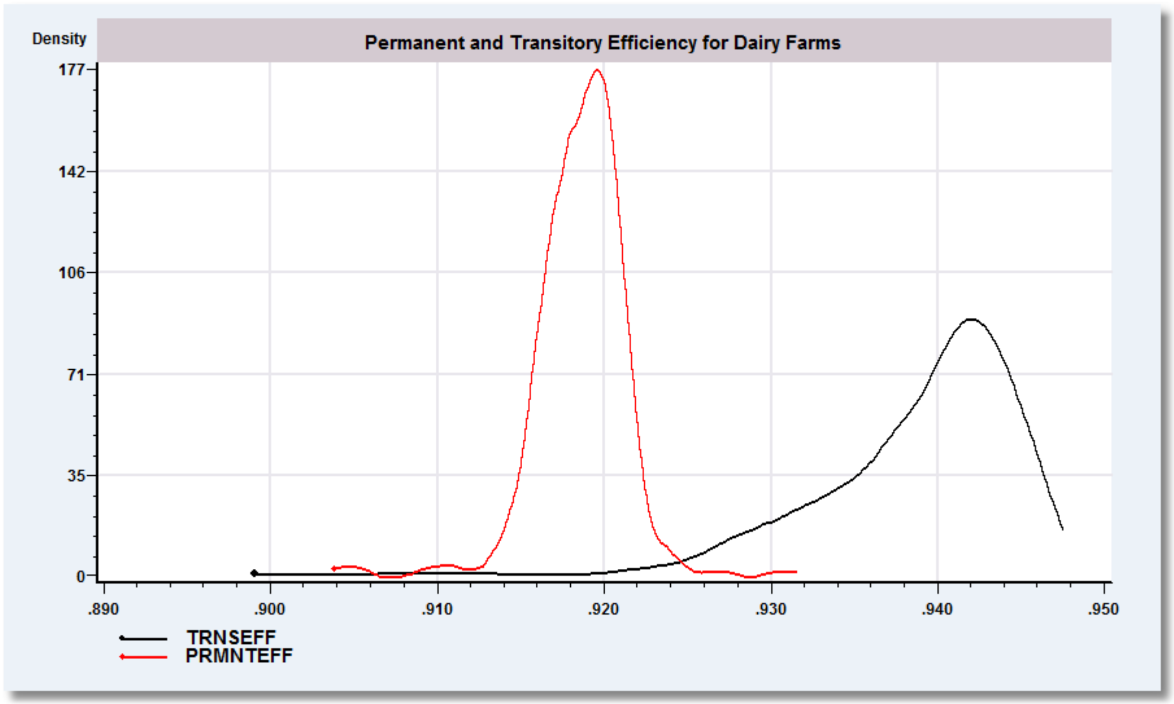
This produces a 'permanent' effect and a 'transitory' effect. The estimator, based on maximum simulated likelihood, has been added to Version 11. (The method was developed in Filippini and Greene (2016).) The following applies the estimator to the panel data used in the preceding example.

```
SETPANEL   ; Group = farm ; Pds = ti $
NAMELIST   ; x = one,x1,x2,x3,x4 $
FRONTIER   ; Lhs = yit ; Rhs = x $
FRONTIER   ; Lhs = yit ; Rhs = x
           ; CSN ; Keep ; Maxit = 25 ; Halton $
CREATE     ; trnseff = Group Mean (eff_sr, Pds = 6) $
CREATE     ; prmnteff = Group Mean (eff_lr, Pds = 6) $
KERNEL     ; If [year98 = 1] ; Rhs = trnseff,prmnteff
           ; Grid
           ; Title = Permanent and Transitory Efficiency for Dairy Farms $
```

```
-----------------------------------------------------------------------------
Random Coefficients  Frontier Model
Dependent variable                  YIT
Log likelihood function      1307.22375
Restricted log likelihood        .00000
Chi squared [  1](P= .000)   2614.44751
Significance level               .00000
Estimation based on N =   1482, K =   9
Inf.Cr.AIC  =  -2596.4 AIC/N =   -1.752
Unbalanced panel has     247 individuals
Simulation  based on   100 Halton draws
---------------------------------------
Closed Skew Normal(LR/SR)Frontier Model
---------------------------------------
---- Short and Long Run Components ----
------- Short Run Time Varying --------
Sigma(uit) (1 sided)   =         .08347
Sigma(vit) (symmetric) =         .06360
--------- Long Run Time Fixed ---------
Theta( ai) (1 sided)   =         .26863
Theta( fi) (symmetric) =         .11475
```

```
--------+--------------------------------------------------------------------
        |                   Standard             Prob.      95% Confidence
     YIT|  Coefficient       Error        z     |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Production / Cost parameters, nonrandom first.......................
      X1|     .63790***      .01003    63.60  .0000       .61824    .65756
      X2|     .03001***      .00624     4.81  .0000       .01778    .04223
      X3|     .03293***      .00752     4.38  .0000       .01820    .04766
      X4|     .40141***      .00542    74.11  .0000       .39079    .41202
        |Means for random parameters.........................................
Constant|    11.6241***      .00519  2237.65  .0000     11.6140   11.6343
        |Theta_fi = std. dev. of time fixed symmetric f(i)...................
Constant|     .11475***      .00212    54.10  .0000       .11059    .11890
        |Variance parameters for v +/- u.....................................
   Sigma|     .10494***      .00297    35.29  .0000       .09911    .11077
        |Asymmetry parameter, lambda.........................................
  Lambda|    1.31243***      .12179    10.78  .0000     1.07373   1.55113
        |Theta_ai = std. dev. of time fixed one sided a(i)...................
Theta_ai|     .26863***      .03432     7.83  .0000       .20136    .33589
--------+--------------------------------------------------------------------
***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
      -------------------------------------------
      Kernel Density Estimator
      Kernel Function     =  Epanechnikov
      Results matrix      =        KERNEL
      Observations        =           247
      Points plotted      =           247
      ------------------------------------------
      Variables Analyzed =       TRNSEFF PRMNTEFF
      Bandwidth           =         .0019    .0008
      Mean                =         .9385    .9186
      Standard Deviation =         .0065    .0026
      Skewness            =       -2.1983   -.9819
      Kurtosis-3 (excess)=        7.3469   7.4499
      Chi2 normality test=       55.0748  46.1383
      Minimum             =         .9010    .9046
      Maximum             =         .9458    .9309
      ------------------------------------------
```



WN1.4 Partially Inefficient Latent Class Stochastic Frontier Model

Kumbhakar, Parmeter and Tsionas (2013) introduced a 'zero inflated' stochastic frontier model which allows an unknown part of the population to operate on the production frontier. The underlying specification is a 'latent class' model in which the data generating process for one class is the usual stochastic frontier model while for the other, there is no inefficiency term, so that for this part, the model is a regression model. *LIMDEP* provides this form of the stochastic frontier model as a form of the latent class stochastic frontier model. The specification can be applied to either cross sections or panels. In the example below, we apply the model to the dairy farms examined above. In fact, based on the results shown, it appears that about half of the farms show no inefficiency term.

```
FRONTIER    ; Lhs = yit ; Rhs = x
            ; Panel ; LCM ; Pts = 2
            ; Rst =  a,b1,b2,b3,b4,s1,lambda1,
                     a,b1,b2,b3,b4,s2,0,p1,p2 $
```

```
-----------------------------------------------------------------------------
Latent Class / Panel Frontier Model
Dependent variable                YIT
Log likelihood function      1081.52238
Restricted log likelihood        .00000
Chi squared [  9](P= .000)   2163.04475
Significance level               .00000
Estimation based on N =   1482, K =   9
Inf.Cr.AIC  =  -2145.0 AIC/N =   -1.447
Unbalanced panel has    247 individuals
Latent class model with 2 latent classes
Stochastic frontier (half normal model)
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Model parameters for latent class 1.................................
Constant|    11.6486***      .00300  3883.74  .0000     11.6427   11.6545
      X1|     .59042***      .00996    59.26  .0000      .57090    .60995
      X2|     .04279***      .00767     5.58  .0000      .02776    .05781
      X3|     .00484         .00894      .54  .5881    -.01268    .02237
      X4|     .43816***      .00457    95.89  .0000      .42921    .44712
   Sigma|     .20829***      .00411    50.71  .0000      .20024    .21634
  Lambda|    4.06030***      .28757    14.12  .0000     3.49668   4.62392
        |Model parameters for latent class 2.................................
Constant|    11.6486***      .00300  3883.74  .0000     11.6427   11.6545
      X1|     .59042***      .00996    59.26  .0000      .57090    .60995
      X2|     .04279***      .00767     5.58  .0000      .02776    .05781
      X3|     .00484         .00894      .54  .5881    -.01268    .02237
      X4|     .43816***      .00457    95.89  .0000      .42921    .44712
   Sigma|     .09835***      .00175    56.23  .0000      .09492    .10178
  Lambda|        0.0    .....(Fixed Parameter).....
        |Estimated prior probabilities for class membership..................
Class1Pr|     .45121***      .04330    10.42  .0000      .36634    .53607
Class2Pr|     .54879***      .04330    12.67  .0000      .46393    .63366
--------+--------------------------------------------------------------------
***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

WN1.5 Data Envelopment Analysis

LIMDEP remains the only package that seamlessly integrates stochastic frontier modeling and data envelopment analysis in the same package. We have continued to develop the DEA component of this package. The bootstrap procedures have been expanded. Bootstrap results have been retained in accessible matrices that can be used in the second step of two step analyses. The general results from DEA, including the bootstrap results are also exportable to *Excel* for further processing.

WN1.6 Fractional Regression for Cross Section and Panel Data

The fractional regression models (for example, Papke and Wooldridge (2008)) are proposed for a dependent variable that is a proportion or is the average of an underlying set of binary responses. *LIMDEP*'s probit and logit estimators have always allowed the dependent variable to be a proportion and modified the likelihood function accordingly. Papke and Wooldridge have extended the specification in several directions, including a panel data specification and allowing for endogenous right hand side variables and random effects. They have also proposed methods of computing appropriate standard errors. Their specification has been built into the program as a new procedure.

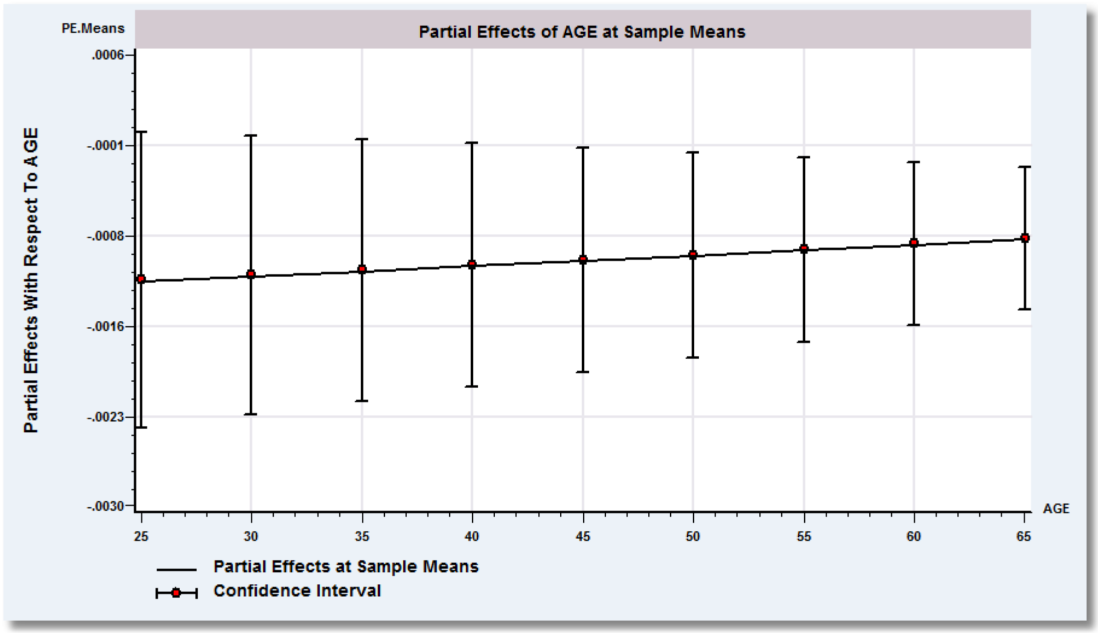
WN1.7 Models with Endogenous Treatment Effects

LIMDEP 11 contains a variant of the linear regression to account for endogenous treatment effects. (It is an extension of the sample selection model in which 'all' observations are selected.) Several other models have been constructed with the same specification, including the main cases, probit, ordered probit, Poisson and negative binomial. In all cases, the model is fit by full information maximum likelihood. For these models, the default function analyzed by **SIMULATE** and **PARTIALS** is the average treatment effect. The following shows an application for a probit model.

```
NAMELIST    ; x = one,age,educ
            ; z = one,age,married,female,income $
PROBIT      ; If [year = 1994] ; Lhs = doctor ; Rhs = x
            ; Treatment = healthy ; Inst = z $
SIMULATE    $
PARTIALS    ; If [year = 1994] ; Effects: age & age = 25(5)65
            ; Plot(c) ; Means $
```

```
-----------------------------------------------------------------------------
Probit with Endogenous Treatment Effect
Dependent variable                DOCTOR
Log likelihood function      -4192.02790
Estimation based on N =   3377, K =  10
Inf.Cr.AIC  =    8404.1 AIC/N =    2.489
--------+--------------------------------------------------------------------
 HEALTHY|                 Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index equation for treatment indicator HEALTHY......................
Constant|    1.10644***      .09375    11.80  .0000      .92269   1.29019
     AGE|    -.02193***      .00195   -11.22  .0000    -.02576   -.01810
 MARRIED|    -.02013         .04181     -.48  .6302    -.10207    .06182
  FEMALE|    -.29251***      .03886    -7.53  .0000    -.36867   -.21635
  INCOME|     .54048***      .09388     5.76  .0000     .35648    .72448
        |Index    equation for DOCTOR........................................
Constant|    1.60653***      .12961    12.40  .0000    1.35251   1.86055
     AGE|    -.00485**       .00223    -2.17  .0297    -.00923   -.00048
    EDUC|     .00123         .00710      .17  .8630    -.01270    .01515
```

```
        |Endogenous treatment indicator.....................................
 HEALTHY|   -1.82020***      .04968   -36.64  .0000    -1.91756  -1.72283
        |Disturbance correlation............................................
Rho(u,e)|     .91439***      .06371    14.35  .0000      .78952   1.03925
--------+--------------------------------------------------------------------
***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------


---------------------------------------------------------------------
Model Simulation Analysis for Probit w/ Endogenous Treatment Effect
Analysis of Average Treatment Effects (ATE)
---------------------------------------------------------------------
Simulations are computed by average over sample observations
---------------------------------------------------------------------
User Function      Function   Standard
(Delta method)      Value      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
Avrg. Function     -.57954     .01424   40.70     -.60745     -.55163

---------------------------------------------------------------------
Partial Effects  Analysis for Probit w/ Endogenous Treatment Effect
Analysis of Average Treatment Effects (ATE)
---------------------------------------------------------------------
Effects on function with respect to AGE
Results are computed at sample means of all variables
Partial effects for continuous AGE      computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dAGE            Partial    Standard
(Delta method)     Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
PE.Func(means)     -.00106     .00048    2.22     -.00200     -.00013
AGE      = 25.00   -.00121     .00061    1.98     -.00241     -.00001
AGE      = 30.00   -.00117     .00058    2.03     -.00230     -.00004
AGE      = 35.00   -.00113     .00054    2.09     -.00220     -.00007
AGE      = 40.00   -.00109     .00051    2.16     -.00208     -.00010
AGE      = 45.00   -.00105     .00047    2.25     -.00196     -.00014
AGE      = 50.00   -.00101     .00043    2.36     -.00184     -.00017
AGE      = 55.00   -.00096     .00038    2.51     -.00171     -.00021
AGE      = 60.00   -.00092     .00034    2.70     -.00158     -.00025
AGE      = 65.00   -.00087     .00029    2.95     -.00145     -.00029
```



WN1.8 Inverse Hyperbolic Sine Model

The log transformation is typically used for analyses of financial variables such as wealth or charitable contributions. In situations in which the data contain extremely large values and/or values very close to zero, logs can produce an unappealing distribution of values. The inverse hyperbolic sine (IHS) transformation,

$$\text{Sinh}^{-1}(y) \;=\; \lambda\text{Sinh}(\lambda\text{y}) = \lambda \ln(\lambda y + (\lambda^2 y^2 + 1)^{1/2})$$

has been proposed as a useful alternative form. (See Burbidge, Magee and Robb (1988), Pence (2006) and Brown, Greene, Harris and Taylor (2015).) Version 11 provides the HIS model with two extensions, a hurdle model and endogenous participation. The hurdle model connects the HIS model to the traditional tobit model – there is an equation for *y* not equal to zero and an intensity equation for observed values of *y* that are strictly positive. The hurdle effect becomes endogenous when there is correlation between the unobservables in the hurdle equation and the unobservables in the main regression equation.

WN1.9 Loglinear Models

Version 10 included a large number of generalized linear and loglinear regression models, including Weibull, inverse gauss, exponential, gamma, binomial, beta (for fractional data), power, geometric, normal, lognormal and Rayleigh. Two new models are added in Version 11, the generalized beta of the second kind and a generalized gamma model. (Jones, Lomas and Rice (2014) is a recent application.) Several interesting special cases of the GB2 model produce a set of additional loglinear models. Like the existing specifications, the new models are provided with access to the panel data applications, fixed and random effects, random parameters and latent classes.

WN2 MAXIMIZE/MINIMIZE and Nonlinear Least Squares

MAXIMIZE/MINIMIZE and **NLSQ** (and **NLSUR**) are used for estimating models with arbitrary functions that you define. You can define your own log likelihood, regression function or system of regression functions. Two major extensions of these procedures have been added in Version 11, panel data functions and random parameters.

WN2.1 Panel Data in MAXIMIZE

MAXIMIZE is used by specifying a term in the log likelihood or other function that is to be summed as the maximand. Panel data functions Sum(…), Prd(…) and Gmn(…) coupled with the panel data setting allow terms that involve within group sums or products. For example, the log likelihood function for a multinomial logit model is

$$\log L = \sum_{i=1}^{n} \log \frac{\Sigma_{j=1}^{J} d_{ij} \exp(\beta' x_{ij})}{\Sigma_{j=1}^{J} \exp(\beta' x_{ij})}.$$

where d_{ij} = 1 if individual *i* makes choice *j* from the set of *J* alternatives. Since the data for a multinomial logit are arranged in blocks of *J* rows of data, we can treat them as a panel. The **MAXIMIZE** command for maximizing this log likelihood function could be

```
NAMELIST    ; xij = the list of variables $
CALC        ; kj = Col(xij) $
MAXIMIZE    ; Labels = kj_b ; Start = kj_0 ; Panel ; Pds = J
            ; Fcn = Num = Sum(dij*Exp(b1’xij)) |
                    Den  = Sum(      Exp(b1’xij)) |
                    Log(num/den) $
```

These would produce the same estimates as the built in command for the multinomial logit model, here

```
CLOGIT      ; Lhs = dij
            ; Choices = the list of J choices
            ; Rhs = variables in xij
            ; Rh2 = one $
```

WN3 New and Expanded Data Descriptive Devices

Several new commands have been added for descriptive statistics. Some of these are components of present programs while a few are new features. The basic **DSTAT** command has been expanded to allow several separate descriptions.

WN3.1 Means

Strata means for stratified data can be obtained separately. In the example below, data on the number of hospital visits are summarized based on the response to the health satisfaction value in the German socioeconomic panel data. The pattern is not surprising, though striking.

```
Doctor Visits by Health Satisfaction
--------------------------+--------------------
DOCVIS      HSAT          |   Obs          Mean
--------------------------+--------------------
          [00] HSAT       |    89       14.5955
          [01] HSAT       |    55       9.83636
          [02] HSAT       |   158       8.76582
          [03] HSAT       |   267       6.95880
          [04] HSAT       |   336       5.33333
          [05] HSAT       |  1034       4.04449
          [06] HSAT       |   630       3.02540
          [07] HSAT       |   981       2.39450
          [08] HSAT       |  1361       1.89934
          [09] HSAT       |   623       1.16212
          [10] HSAT       |   675       1.24148
--------------------------+--------------------
          DOCVIS          |  6209       3.13400
--------------------------+--------------------
```

WN3.2 Correlations

Correlation matrices are requested separately with a simple instruction,

CORRELATION ; Rhs = the list of variables $

WN3.3 Tables

Tables of descriptive statistics by strata are requested with

TABLE ; Rhs = the variable ; Str = the stratification indicator $

For example,

TABLE ; Rhs = income ; Str = year $

```
---------+------------------------------------------------------------
         | Means and Standard Deviations for Clustered  or Stratified Data
         | Variable  =  INCOME      Weights for observations are 1.000000
         | Full Sample =   27326 data rows. Valid rows =    27326
         | Rows skipped (bad stratum or weight)        =        0
         | Sum of weights for all valid observations   =    27326.000
 Overall | Mean        Std Dev.   Sum of Weights   Sample  Missing    Total
         |      .352        .177     27326.000      27326        0    27326
 Stratum | There were    7 strata   found in the sample
         | Mean        Std Dev.   Sum of Weights   Sample  Missing    Total
---------+------------------------------------------------------------
Stratum00|      .297        .148      3874.000       3874        0     3874
Stratum01|      .309        .140      3794.000       3794        0     3794
Stratum02|      .325        .165      3792.000       3792        0     3792
Stratum03|      .349        .164      4483.000       4483        0     4483
Stratum04|      .336        .158      3666.000       3666        0     3666
Stratum05|      .407        .191      4340.000       4340        0     4340
Stratum06|      .445        .217      3377.000       3377        0     3377
---------+------------------------------------------------------------
```

WN3.4 Quantiles

Quantiles for a list of variables are obtained with

QUANTILES ; Rhs = the list of variables $

For example, the following shows the results for two variables in the health data set. The .025 and .975 values are highlighted. In some applications involving bootstrapping, the quantiles will be used for computing confidence intervals.

```
---------------------------------------
Percentiles               AGE     INCOME
Sample size             27326      27326
---------------------------------------
Min.                     25.0      .0015
01th                     25.0        .08
*025                     25.0       .105
05th                     26.0        .14
10th                     28.0      .1789
20th                     32.0       .214
25th                     34.0        .24
30th                     36.0        .25
40th                     39.0         .3
Med.                     43.0        .32
60th                     47.0        .36
70th                     51.0         .4
75th                     53.0        .43
80th                     55.0        .46
90th                     60.0        .55
95th                     62.0        .65
*975                     63.0     .77375
99th                     64.0        .93
Max.                     64.0     3.0671
---------------------------------------
```

WN3.5 Frequencies

Frequencies for a discrete variable are obtained with

FREQUENCIES ; Rhs = the variable $

The sample of values on health satisfaction in the German health care data are found with

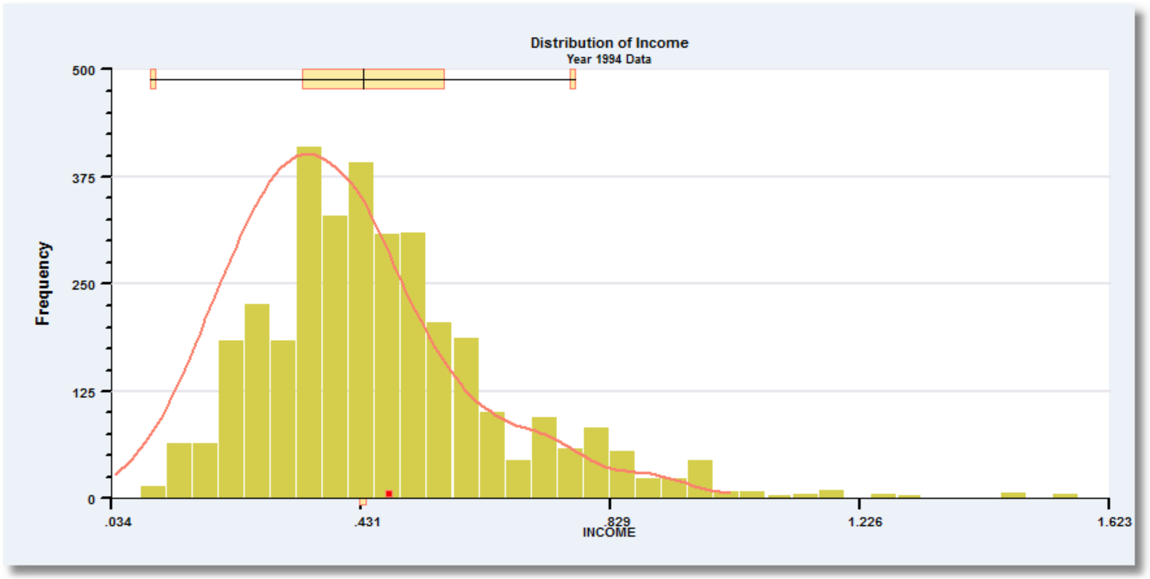
```
-------------------------------------
Frequency Table for      HSAT
Sample size             27326
-------------------------------------
                Sample       Sample
   Value       Frequency   Proportion
-------------------------------------
    HSAT=    0       446         .0163
    HSAT=    1       254         .0093
    HSAT=    2       641         .0235
    HSAT=    3      1172         .0429
    HSAT=    4      1389         .0508
    HSAT=    5      4232         .1549
    HSAT=    6      2529         .0925
    HSAT=    7      4230         .1548
    HSAT=    8      6171         .2258
    HSAT=    9      3060         .1120
    HSAT=   10      3191         .1168
-------------------------------------
```

WN4 Extensions of Graphical Descriptive Devices

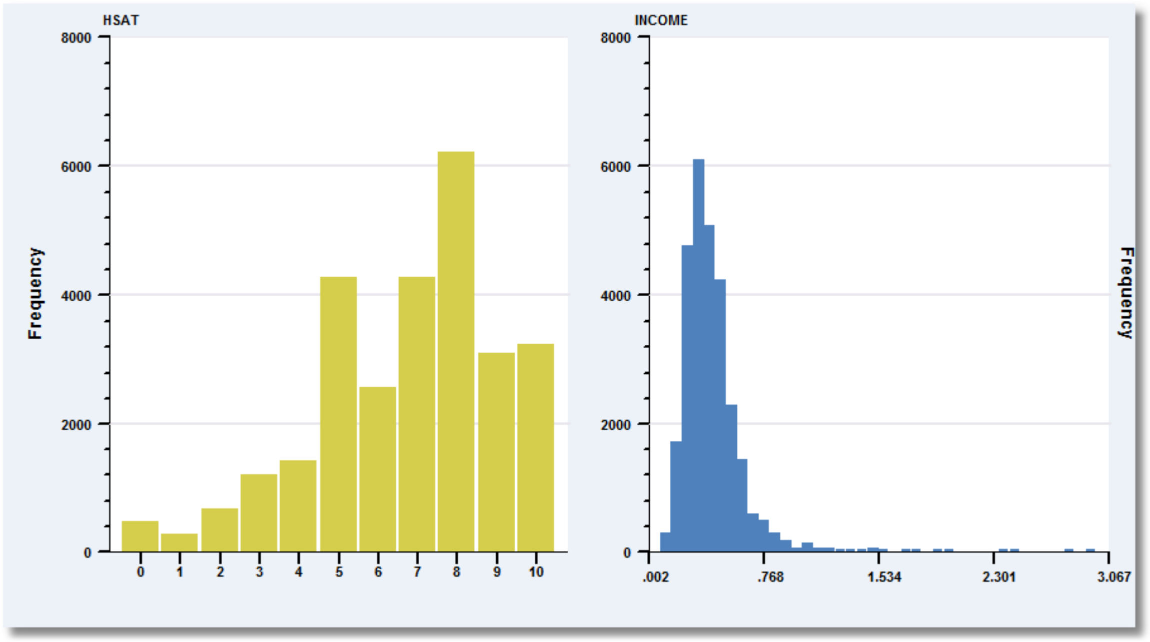
Several graphical devices are provided for describing data. Options have been added to allow a greater variety of figures. The extensions include addition of titles, subtitles and subtitles, templates for legends in figures and selection of a palette of pen colors, styles and weights. Features have been added for the specific graphics, including box plots, histograms and kernel density estimators.

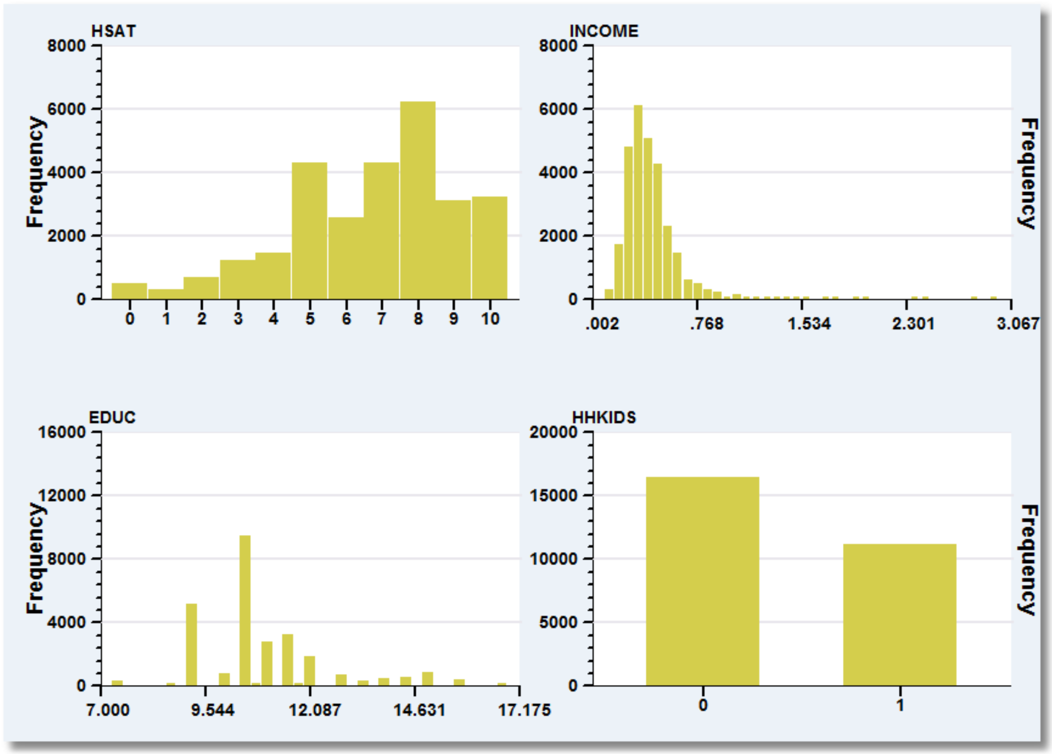
WN4.1 Histograms

Histograms can be overlaid with box plots and/or kernel density estimators, as in the example below that describes an income distribution for one year of a panel.

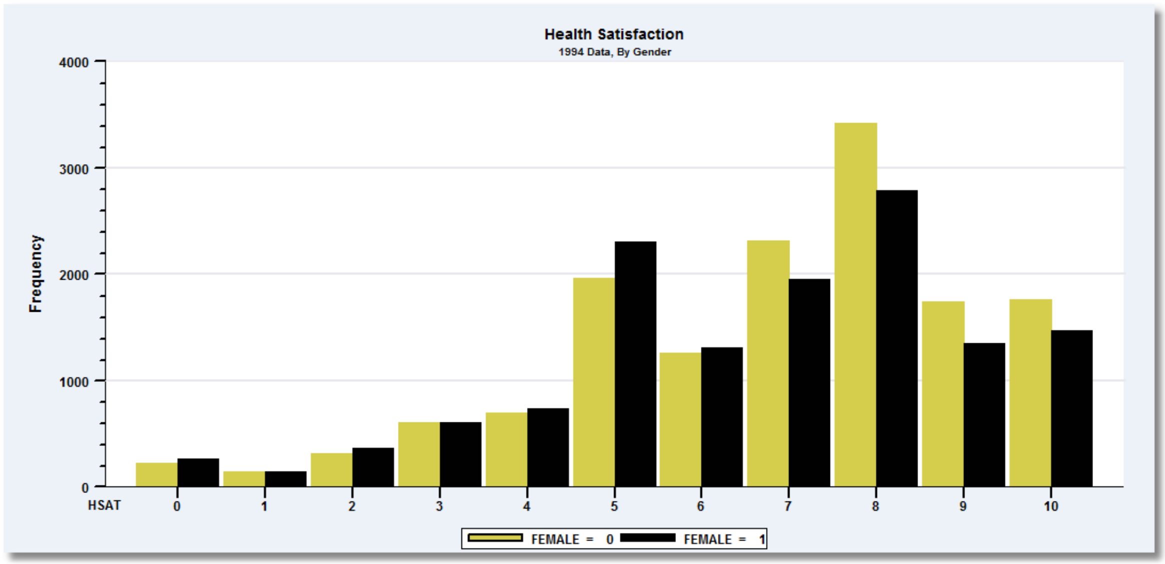


Up to four histograms can be in grouped in a single arrangement:



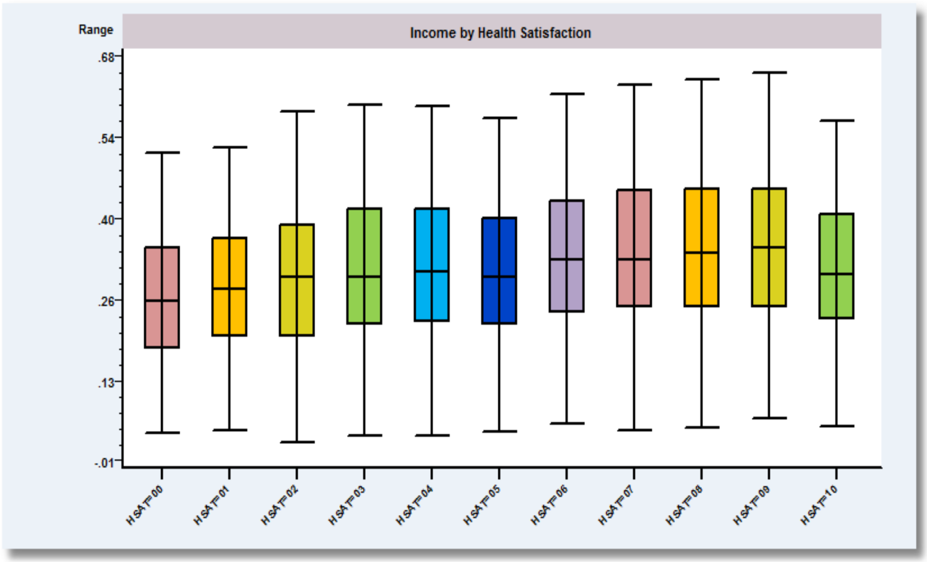


Histograms can be grouped for comparisons of up to five groups.



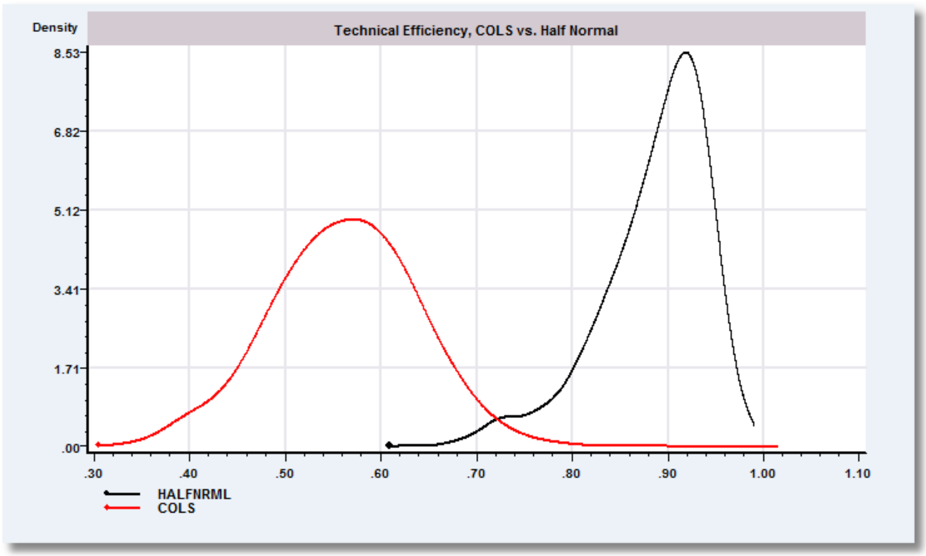
WN4.2 Box Plots

Several different specifications have been added for boxplots. The limit on the number of plots in a figure has been increased to 40. A specific command, **BOXPLOTS** is used to request this type of figure.



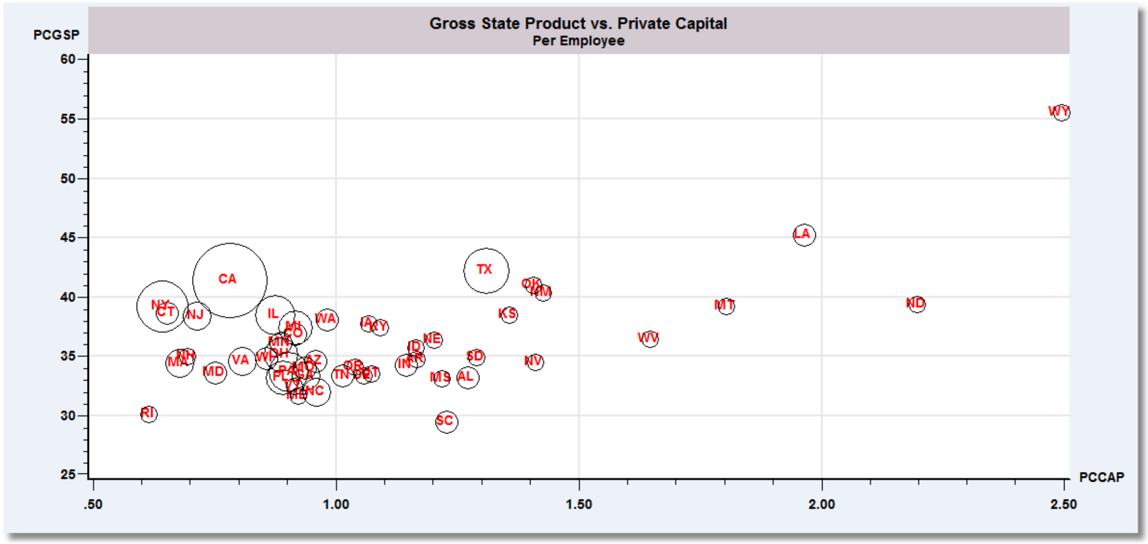
WN4.3 Kernel Density Estimator

Several new specifications are provided for plotting kernel density estimators. The following compares the estimates of technical efficiency from two specifications of a stochastic frontier model.



WN4.4 Bubble Plot

A bubble plot is essentially a scatter plot with unequal weights displayed as the diameters of the bubble for the observations.

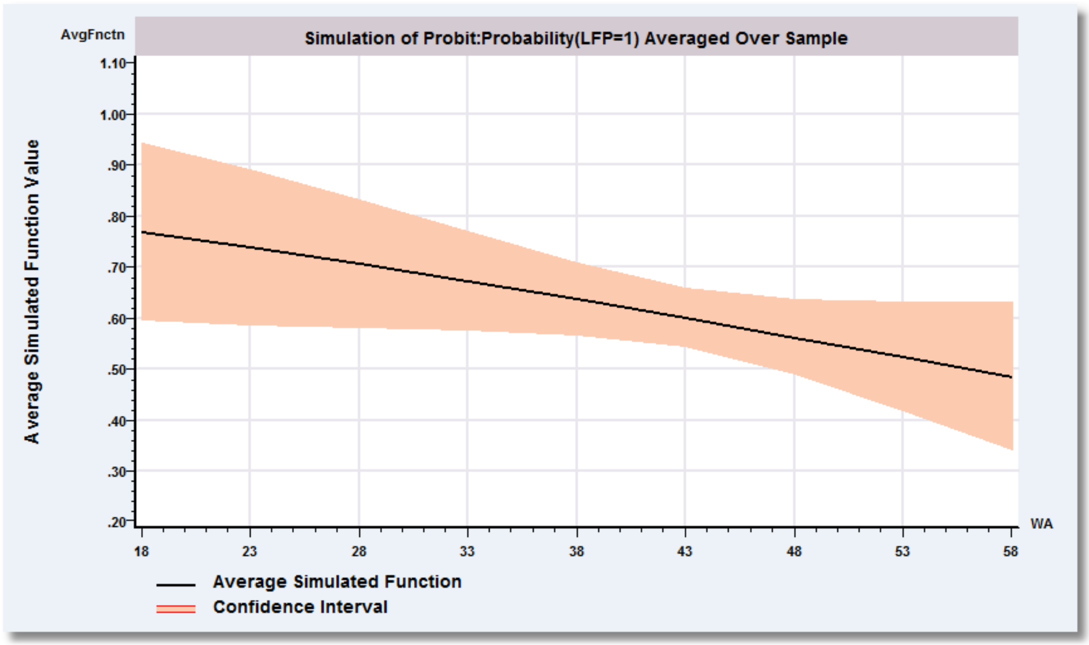


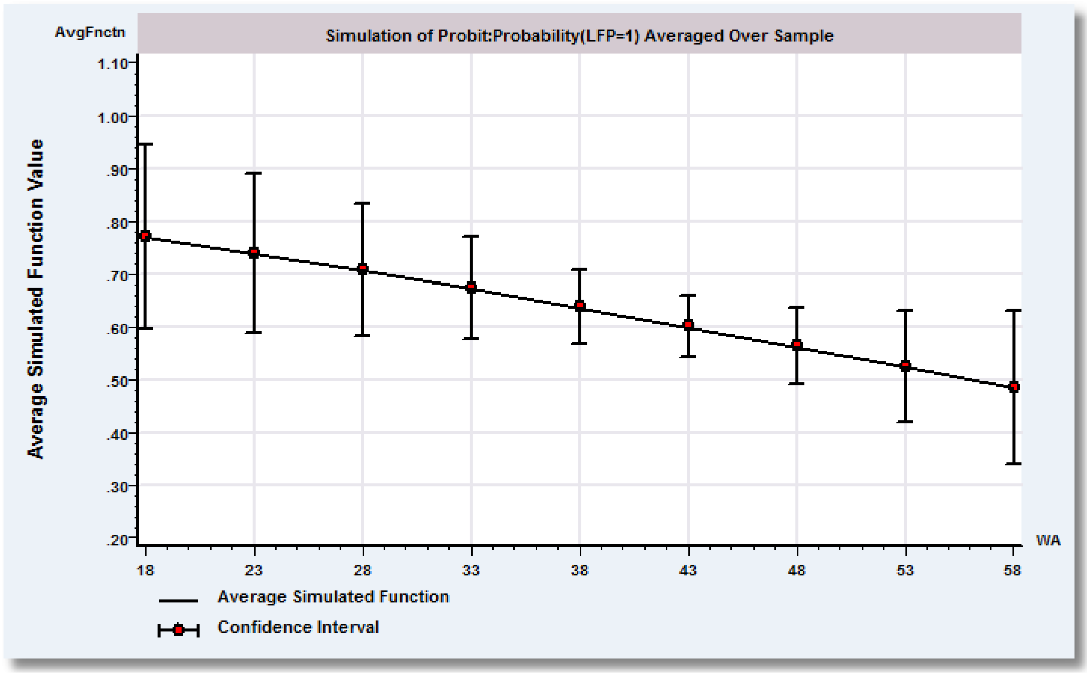
WN5 PARTIALS and SIMULATE

PARTIALS and **SIMULATE** are used to analyze the implications of an estimated model. Nonlinearities and interactions make it necessary to examine the model more closely after estimation to go beyond simple signs and significance as the outputs of estimation. **SIMULATE** is used to predict the outcome function for a model. **PARTIALS** is used to examine the partial effects of variables that appear in the model while accounting for the many parts of a nonlinear model. Simple coefficients are almost always inadequate to provide this information. These two ‘post estimation’ tools were introduced for most models in Version 10. We have continued to expand the list of models included with this capability.

WN5.1 Graphical Devices

Plots of simulated values from the estimated model and plots of the behavior of partial effects are useful devices for learning the implications of a model. Plotting confidence intervals with the estimated values provides additional information about the implied estimates from a nonlinear model. The following show two different formats for the confidence regions for the partial effects from a probit model.


AvgFnctn
Simulation of Probit:Probability(LFP=1) Averaged Over Sample
Average Simulated Function Value
1.10
1.00
.90
.80
.70
.60
.50
.40
.30
.20
WA
18
23
28
33
38
43
48
53
58
Average Simulated Function
Confidence Interval


AvgFnctn
Simulation of Probit:Probability(LFP=1) Averaged Over Sample
Average Simulated Function Value
1.10
1.00
.90
.80
.70
.60
.50
.40
.30
.20
WA
18
23
28
33
38
43
48
53
58
Average Simulated Function
Confidence Interval

WN5.2 Transition Matrices for Partial Effects of Categorical Variables

In the following probit model, *edlevel* is a categorical variable, coded 0,1,2,3 for four levels of education, *lths* (less than high school), *highschl*, *college*, and *grad*.

PROBIT ; Lhs = doctor ; Rhs = one,age,income, #edlevel $

```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable                  DOCTOR
Log likelihood function       -17697.13404
Restricted log likelihood     -18019.55173
Chi squared [  5](P= .000)       644.83539
Significance level                  .00000
McFadden Pseudo R-squared         .0178927
Estimation based on N =  27326, K =   6
Inf.Cr.AIC  =  35406.3 AIC/N =    1.296
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability......................................
Constant|    -.30317***      .03575    -8.48  .0000     -.37324   -.23310
     AGE|     .01606***      .00071    22.46  .0000      .01466    .01746
  INCOME|    -.13615***      .04534    -3.00  .0027     -.22502   -.04728
--------+--------------------------------------------------------------------
 EDLEVEL| Base = 0
      1 |     .05327**       .02124     2.51  .0122      .01163    .09490
      2 |    -.05341*        .02727    -1.96  .0502     -.10687    .00004
      3 |    -.23230***      .03157    -7.36  .0000     -.29418   -.17042
--------+--------------------------------------------------------------------
***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Partial effects based on this model would typically examine the effect of switching from the base level (less than high school) to each of the other levels. One might be interested in a different experiment, say for example, the effect of a switch from level 1 (high school) to level 2 (college). In principle, this could be deduced from the original set of three partial effects, though for a detailed analysis, this is likely to become rather inconvenient. The full set of transitions is automated in **PARTIALS**, as shown in the results below. The transition to ‘(Other)’ corresponds to the conceptual experiment of the individual switching out of the given category, but not to a specific other category – that is, to a weighted average of the other categories.

```
PARTIALS    ; Effects: #edlevel
            ; Transition
            ; Category = lths,highschl,college,grad $
```

```
----------------------------------------------------------------------
Simulation and partial effects based on categorical variable EDLEVEL
Results computed by setting all observations to category value and
comparing to base value.
Sample proportions apply to full sample before @ settings in command
----------------------------------------------------------------------
Category Dummy  Sample | Fraction  Category
Base value   0   17944    .65666   LTHS
             1    4909    .17965   HIGHSCHL
             2    2597    .09504   COLLEGE
             3    1876    .06865   GRAD
----------------------------------------------------------------------
Partial Effects  Analysis for Probit:Probability(DOCTOR=1)
----------------------------------------------------------------------
Effects of switches between categories in EDLEV=xx (dummy variables)
Results are computed by average over sample observations
HIGHSCHL= .1796   COLLEGE = .0950   GRAD    = .0687   LTHS    = .6567
----------------------------------------------------------------------
df/dEDLEV=xx       Partial    Standard
From --> To        Effect      Error     |t|  95% Confidence Interval
----------------------------------------------------------------------
HIGHSCHL HIGHSCHL   .00000     .00000     .00      .00000      .00000
HIGHSCHL COLLEGE  -.03947     .01150    3.43     -.06202     -.01692
HIGHSCHL GRAD     -.10816     .01326    8.16     -.13414     -.08217
HIGHSCHL LTHS     -.01953     .00774    2.52     -.03470     -.00436
HIGHSCHL (Other)  -.02926     .00749    3.91     -.04393     -.01458
COLLEGE  HIGHSCHL   .03947     .01150    3.43      .01692      .06202
COLLEGE  COLLEGE    .00000     .00000     .00      .00000      .00000
COLLEGE  GRAD     -.06869     .01480    4.64     -.09769     -.03969
COLLEGE  LTHS       .01994     .01024    1.95     -.00014      .04001
COLLEGE  (Other)    .01709     .00997    1.71     -.00246      .03664
GRAD     HIGHSCHL   .10816     .01326    8.16      .08217      .13414
GRAD     COLLEGE    .06869     .01480    4.64      .03969      .09769
GRAD     GRAD       .00000     .00000     .00      .00000      .00000
GRAD     LTHS       .08863     .01226    7.23      .06460      .11265
GRAD     (Other)    .09036     .01200    7.53      .06685      .11387
LTHS     HIGHSCHL   .01953     .00774    2.52      .00436      .03470
LTHS     COLLEGE  -.01994     .01024    1.95     -.04001      .00014
LTHS     GRAD     -.08863     .01226    7.23     -.11265     -.06460
LTHS     LTHS       .00000     .00000     .00      .00000      .00000
LTHS     (Other)  -.01302     .00639    2.04     -.02554     -.00051
----------------------------------------------------------------------
* (Other) = conditional share weighted average of all switch effects
----------------------------------------------------------------------
Partial Effects Transition Matrix for EDLEV=xx
There are  4 categories (sample %)
  01=HIGHSCHL (17.96) 02=COLLEGE  ( 9.50) 03=GRAD     ( 6.87)
  00=LTHS     (65.67)
Entry = effect on outcome of switch from row category to column
Switch to Other is unspecified switch out of row category
--------+-----------------------------------------------------
        |     01     02     03     00  Other
--------+-----------------------------------------------------
HIGHSCHL|   .000  -.039  -.108  -.020  -.029
 COLLEGE|   .039   .000  -.069   .020   .017
    GRAD|   .108   .069   .000   .089   .090
    LTHS|   .020  -.020  -.089   .000  -.013
--------+-----------------------------------------------------
```

WN5.3 Krinsky and Robb Method

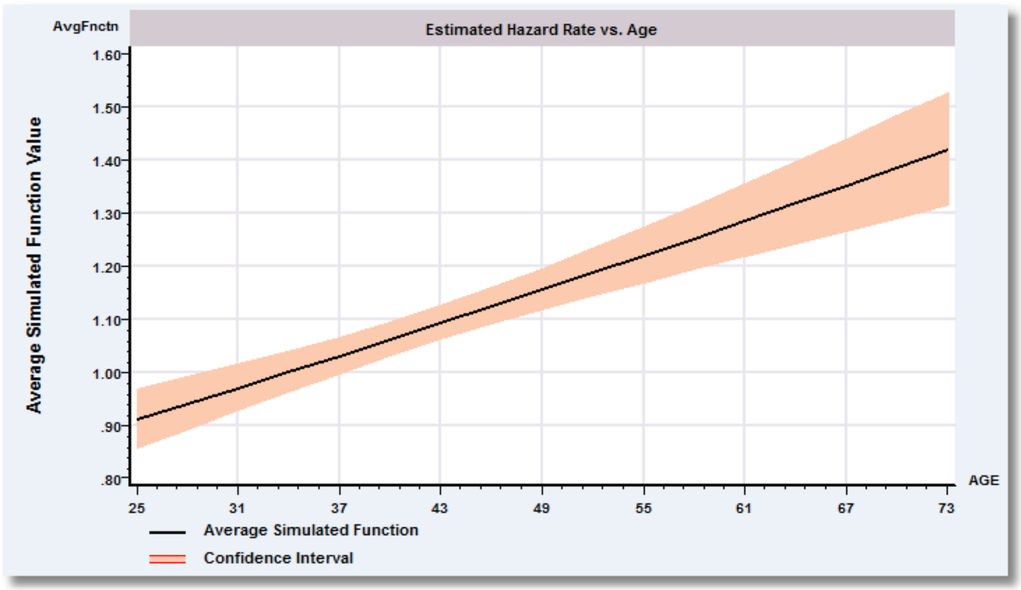
Standard errors for partial effects (at the means or averaged across observations) are typically computed using the delta method. In some cases, the method of Krinsky and Robb might be preferred. *LIMDEP* Version 11 allows the choice of either method for computing standard errors for simulations or for partial effects.

WN5.4 Customized Models and Extensions

The **SIMULATE** and **PARTIALS** commands are more general than the typical application would suggest. Either processor allows the model on which the computations are based to be either a default specification from the most recent estimation (typically a mean function or a probability), or any other linear or nonlinear function that the user specifies using the program’s nonlinear modeling procedures. For example, the following estimates a probit model. The default function for **SIMULATE** and **PARTIALS** would be the predicted probability. The following examines the hazard function for the estimated model:

```
NAMELIST    ; x = one,age,educ,female,married,income $
CALC        ; k = Col(x) $
PROBIT      ; If [year = 1994] ; Lhs = doctor ; Rhs = x $
SIMULATE    ; If [year = 1994]
            ; Parameters = b
            ; Covariance = varb
            ; Labels = k_b
            ; Function = N01(b1'x)/(1 - Phi(-b1'x))
            ; Scenario: & age = 25(3)75
            ; Plot(ci) $
```

```
---------------------------------------------------------------------
Model Simulation Analysis for User Specified Function
---------------------------------------------------------------------
Simulations are computed by average over sample observations
---------------------------------------------------------------------
User Function      Function   Standard
(Delta method)      Value      Error      |t|  95% Confidence Interval
---------------------------------------------------------------------
Avrg. Function     1.09168     .01632   66.90     1.05969     1.12366
AGE      = 25.00    .91173     .02844   32.06      .85599      .96747
AGE      = 28.00    .94101     .02528   37.22      .89145      .99057
AGE      = 31.00    .97066     .02231   43.51      .92693     1.01439
AGE      = 34.00   1.00067     .01967   50.86      .96211     1.03923
AGE      = 37.00   1.03103     .01761   58.56      .99652     1.06554
AGE      = 40.00   1.06173     .01641   64.70     1.02956     1.09389
AGE      = 43.00   1.09276     .01636   66.80     1.06069     1.12482
AGE      = 46.00   1.12411     .01754   64.10     1.08974     1.15848
AGE      = 49.00   1.15578     .01979   58.41     1.11699     1.19456
AGE      = 52.00   1.18775     .02285   51.98     1.14296     1.23254
AGE      = 55.00   1.22002     .02649   46.06     1.16811     1.27194
AGE      = 58.00   1.25259     .03053   41.03     1.19275     1.31242
AGE      = 61.00   1.28543     .03487   36.87     1.21709     1.35377
AGE      = 64.00   1.31855     .03943   33.44     1.24127     1.39583
AGE      = 67.00   1.35193     .04417   30.61     1.26536     1.43850
AGE      = 70.00   1.38557     .04905   28.25     1.28943     1.48171
AGE      = 73.00   1.41947     .05406   26.26     1.31352     1.52542
```



WN6 Analysis Tools and Capabilities

Three tools are used for directly manipulating data and statistics, **CALCULATE**, **CREATE** and **MATRIX**. Many new functions have been added for all three. **CREATE** in particular provides over 30 new functions for panel data transformations, including, for example,

CREATE ; yinitial = Group Obs1 (yit, Pds = ti) $

which creates the new variable that is equal to the value of the original variable in the first period. Use this transformation to include the initial value in the set of variables in a dynamic panel data model.

New random number generators have been added for skew normal, negative binomial (P) and inverse Gauss. Other new functions include the minimum or maximum of a variable.

CALCULATE and **MATRIX** have also been extended by dozens of new functions. The matrix functions include several new computations for panel data.

WN7 Programming with EXECUTE and PROCEDURES

EXECUTE may be used to iterate over subsets of the data, for example, for testing heterogeneity over strata or groups defined by the observation tags. Several new specifications are provided for defining bootstrap and jackknife iterations, including a bias correction for the jackknife estimator.

WN8 Importing and Using Character Variables

LIMDEP 11 adds several methods of using character data related to your data analysis. The character data come in several forms:

1. Observation tags which are variables that contain characters rather than numeric values
2. Labellists
3. A roster of up to 50 80 byte character strings that are used for several purposes

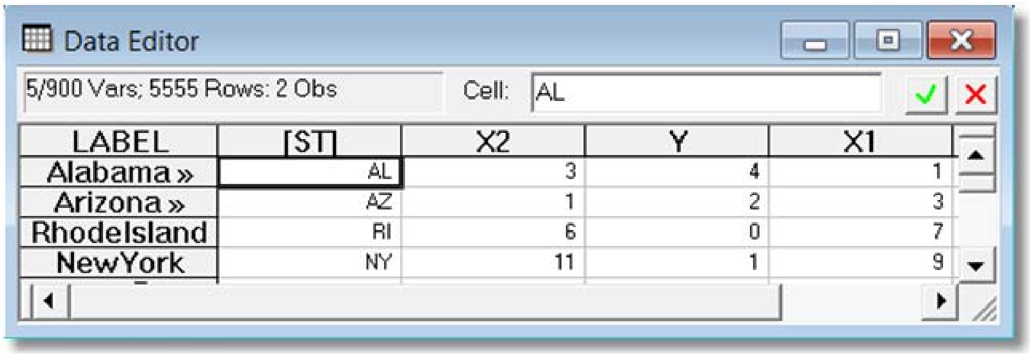
Your input data may contain a variable that is a character name rather than a numeric. Here is a small example of a mixed data set that contains both observation labels (at the left) and a tag variable.

```
Import ; tags = ST $
Label        x1  x2  y  ST
Alabama       1   3  4  AL
Arizona       3   1  2  AZ
RhodeIsland   7   6  0  RI
NewYork       9  11  1  NY
```

The observation labels are used by the program in the data editor and in some lists of data such as fitted values. The tag variable can be used in several settings, such as setting the sample:

INCLUDE ; New ; [ST] = AL,AZ $

The result of importing these data, then issuing the **INCLUDE** command is shown in Figure WN1.



LABEL	[ST]	X2	Y	X1
Alabama »	AL	3	4	1
Arizona »	AZ	1	2	3
RhodeIsland	RI	6	0	7
NewYork	NY	11	1	9

Figure WN1 Observation Tags Used with INCLUDE

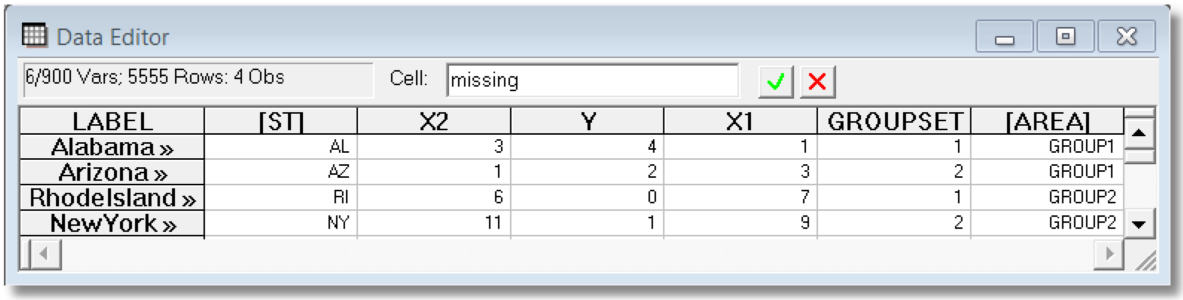
Tags may be used with **INCLUDE**, **REJECT** and **SAMPLE** to control the current sample.

The observation tags may be used in several settings, include **CREATE**, executing procedures and in general model commands.

The tags may be aggregated into strata. For example, in the preceding,

CREATE ; [Area] = Groups ([ST]): Group1 (AL,AZ) / Group2 (RI,NY) $

aggregates the four observations into two strata. The data set now contains



6/900 Vars; 5555 Rows: 4 Obs Cell: missing

LABEL	[ST]	X2	Y	X1	GROUPSET	[AREA]
Alabama »	AL	3	4	1	1	GROUP1
Arizona »	AZ	1	2	3	2	GROUP1
RhodeIsland »	RI	6	0	7	1	GROUP2
NewYork »	NY	11	1	9	2	GROUP2

Figure WN2 Aggregated Observation Tags

(It is possible to aggregate numeric data into alphabetic tags, or vice versa, to produce easily manipulated strata in the data.) Observation tags can be used to control execution in the same fashion as the numeric data. For example, the sample for this command is controlled by

```
DSTAT       ; If  [[AREA] = Group1] Rhs = x1,x2 $
```

which produces

```
--------+-------------------------------------------------------------------
        |              Standard                                      Missing
Variable|         Mean    Deviation       Minimum       Maximum    Cases  Values
--------+-------------------------------------------------------------------
      X1|          2.0     1.414214           1.0           3.0        2       0
      X2|          2.0     1.414214           1.0           3.0        2       0
--------+-------------------------------------------------------------------

Descriptive Statistics for    2 variables
DSTAT results are matrix LASTDSTA in current project.
```

It is also possible to use the strata in iterations, for example,

```
REGRESS     ; For [[AREA] = Group1,Group2] ; Lhs = y ; Rhs = x1 $
```

```
----------------------------------------------------
Setting up an iteration over the values of [AREA]
The model command will be executed for     2 values
of this variable.  In the current sample of       4
observations, the following counts were found:
Subsample  -Observations   Subsample  -Observations
[AREA]   = GROUP1      2   [AREA]   = GROUP2      2
----------------------------------------------------
Actual subsamples may be smaller if missing values
are being bypassed.  Subsamples with 0 observations
will be bypassed.
----------------------------------------------------
Subsample analyzed for this command is [AREA]   = GROUP1
```

The iteration over strata can be used for a homogeneity test, as in

```
REGRESS     ; For [(test) [AREA]] ; Lhs = y ; Rhs = x1 $
```

This iteration does the computation for all of the subsamples defined by [AREA] and tests for homogeneity using an F test for regression or a likelihood ratio test for other models. Iterations over strata may be defined in **EXECUTE** as well. For example,

```
PROCEDURE $
REGRESS     ; Lhs = y ; Rhs = one,x1,x2 $
ENDPROCEDURE $
EXECUTE     ; [AREA] = Group1,Group2 $
```

(Of course, with two observations in each group, the regression is not actually computable.)

Direct manipulations of the data using **CALC**, **CREATE** and **MATRIX** may also be controlled with the observation tags, as in, for example,

```
NAMELIST    ; x = one,x1,x2 $
MATRIX      ; For [[AREA] = Group1] ; xx = x'x $
```

This uses the observations in Group 1 to compute the matrix. This sort of construction can also be used in **CREATE**, **RECODE** and **CALC** when those commands are used to manipulate the sample operations.

The observation tags can be mapped to a categorical variable. For example, we suppose in the data that education level is labeled *lths*, *hs*, *coll*, *grad*. The data may be used as follows:

```
CREATE      ; edlevel = Groups([ED]): 1(lths)/2(hs)/3(coll)/4(grad) $
REGRESS     ; Lhs = income ; Rhs = one,#edlevel,age,exper $
```

WN9 Missing Data

The program default code for missing values is -999. In general, one would not find it necessary to change this. In importing data, missing values are typically indicated by some nonnumeric value, such as '.' or an alphabetic indicator such as 'missing.' Rarely, a data set may have been created with some other specific value to indicate missing (such as -99). You may now redefine the default missing value at the time you import your data with

```
IMPORT      ; … ; Missing value = the value $
or READ
```

or later with

```
CALC        ; Missing value = the value $
```

In general, it is not necessary ever to refer to the specific value. **SKIP** and various data manipulations handle that internally. For **REJECT** and **INCLUDE**, for example, you can use '.' to indicate missing, as in

```
REJECT     ; x = . $
```

If you are not using imputation or some other method to fill missing values, your model estimation will use 'listwise deletion.' Observations will be skipped if any variable in the list is missing. You can now set the current sample specifically to what will be the resulting setting as follows:

```
NAMELIST   ; name  =  … the full list $
REJECT     ; New ; name = . $
```

The **SKIP** command accomplishes this setting during estimation, but does not reset the current sample. The listwise deletion is only in place when the model is estimated. It might be useful to use the resulting sample in a subsequent operation. After each model command, a binary variable named *_sample_* is created that equals one for observations that were used in the previous model and zero if not. For example, suppose the three variables *x*1, *x*2 and *x*3 all contain missing values, in different, perhaps overlapping places. The sample used can be maintained as follows:

```
SKIP
REGRESS    ; Lhs = y ; Rhs = one,x1,x2,x3 $
DSTAT      ; If [ _sample_ = 1] ; Rhs = x1, x2, x3 $
```

The **DSTAT** command will report the results for the three variables using the sample that was used to compute the regression. Note that the results will be different from what would result if the **; If […]** function were omitted because **DSTAT** will otherwise use all the available data for each variable in turn.

R1: Introduction to *LIMDEP* Version 11

The documentation for *LIMDEP* is divided into two parts: this *Reference Guide* and a separate *Econometric Modeling Guide*. The *LIMDEP Reference Guide* describes how to use *LIMDEP* to read a data set, establish the current sample, compute transformations of variables, and carry out other functions that get your data ready to use for estimation purposes. Several important tools, such as the matrix algebra program, scientific calculator and program editor are described here as well. The *LIMDEP Reference Guide* also describes general features of the program used in most model frameworks, including computing partial effects, general tools for testing hypotheses, panel data models and using multiple imputation for filling missing data. The second part of the manual, the *Econometric Modeling Guide* describes specific modeling frameworks and instructions to be used for fitting these models. For *NLOGIT* Version 6 users, there is a separate manual, the *NLOGIT Reference Guide* dedicated to the special features of *NLOGIT*.

R1.1 The *LIMDEP* Program

LIMDEP is oriented toward cross section and panel data. But, many standard techniques for time series analysis are supported as well. *LIMDEP*'s basic procedures for data analysis include:

- descriptive statistics (means, standard deviations, minima, etc.), with stratification,
- kernel density estimation, histograms, box plots and other broad descriptive tools,
- univariate tests such as equality of means and tests for pooling,
- cross tabulations and scatter plots of several types,
- multiple linear regression,
- nonparametric regression,
- time series identification, autocorrelations and partial autocorrelations,

You can also model the sorts of extensions of the linear regression model normally needed for teaching and research, such as:

- heteroscedasticity with robust standard errors,
- autocorrelation with robust standard errors,
- multiplicative heteroscedasticity,
- groupwise heteroscedasticity and cross sectional correlation,
- the Box-Cox regression model,
- one and two way random and fixed effects models for balanced or unbalanced panel data,
- distributed lag models, ARIMA, and ARMAX models,
- time series models with GARCH effects,
- dynamic linear models for panel data,
- nonlinear single and multiple equation regression models,
- seemingly unrelated linear and nonlinear regression models,
- simultaneous equations models.

LIMDEP is best known for its extensive menu of programs for estimating the parameters of nonlinear models for qualitative and limited dependent variables. (We take our name from *LIM*ited *DEP*endent variables.) No other package supports a greater variety of nonlinear econometric models. Among *LIMDEP*'s more advanced features, each of which is invoked with a single command, are:

- univariate, bivariate and multivariate probit models, probit models with partial observability, sample selection, heteroscedasticity and random effects,
- Poisson and negative binomial models for count data, with fixed or random effects, sample selection, underreporting, and numerous other models of over- and underdispersion,
- two part models such as hurdle and zero inflation,
- tobit and truncation models for censored and truncated data,
- models of sample selection with one or two selection criteria,
- parametric and semiparametric duration models with time varying covariates,
- stochastic frontier regression models and data envelopment analysis,
- ordered probit and logit models, with censoring and sample selection,
- switching regression models,
- nonparametric and kernel density regression,
- fixed effects models, random parameters models and latent class models for over 40 different linear and nonlinear model classes,

and over fifty other model classes. Each of these allows a variety of different specifications. Most of the techniques in wide use are included. Among the aspects of this program which you will notice early on is that regardless of how advanced a technique is, the commands you use to request it are the same as those for the simplest regression.

NLOGIT Version 6 includes all the features of *LIMDEP* Version 11 and offers in addition:

- FIML estimation of nested logit models with up to four levels including several formats that build in assumptions of utility maximization,
- LIML estimation of conditional and multinomial logit models,
- heteroscedastic extreme value models,
- covariance heterogeneity in nested logit models,
- random parameters logit models for cross sections and panel data,
- multinomial probit and multiperiod multinomial probit models for panel data,
- generalized nested logit models with overlapping nests,
- kernel logit models with several different formats of individual effects,
- heteroscedastic random parameters models,
- the random regret form of the multinomial logit model,
- best/worst specifications for multinomial logit,
- multinomial choice models with nonlinear utility functions,
- numerous forms of latent class multinomial logit models, including random parameters,
- generalized mixed logit models.

LIMDEP also provides numerous programming tools, including an extensive matrix algebra package and a function optimization routine, so that you can specify your own likelihood functions and add new specifications to the list of models. All results are kept for later use. You can use the matrix program to compute test statistics for specification tests or to write your own estimation programs. (The manual contains numerous examples.) The structure of *LIMDEP*'s matrix program is also especially well suited to the sorts of moment based specification tests suggested, for example, in Pagan and Vella (1989) – all the computations in this paper were done with *LIMDEP*. The programming tools, such as the editor, looping commands, data transformations, and facilities for creating 'procedures' consisting of groups of commands will also allow you to build your own applications for new models or for calculations such as complicated test statistics or covariance matrices. A new package of programs allows analysis of partial effects of any number of interaction terms and any degree of complexity, for any variable in any model, or in any modification of a model that you can formulate yourself.

Most of your work will involve analyzing data sets consisting of externally generated samples of observations on a number of variables. You can read the data, transform them in any way you like, for example, compute logarithms, lagged values, or many other functions, edit the data, and, of course, apply the estimation programs. You may also be interested in generating random (Monte Carlo) samples rather than analyzing 'live' data. *LIMDEP* contains random number generators for 20 discrete and continuous distributions including normal, truncated normal, Poisson, discrete or continuous uniform, binomial, logistic, Weibull, and others. A facility is also provided for random sampling or bootstrap sampling from any data set, whether internal or external, and for any estimation technique you have used, whether one of *LIMDEP*'s routines or your own estimator created with the programming tools. *LIMDEP* also provides a facility for bootstrapping panel data estimators.

R1.2 Econometric Techniques

This manual is devoted to use of this program. As such, there is relatively little instructional material on the econometric models and techniques. Where possible, we have included sources to refer to and a small amount of background material. For those users not already experienced in empirical econometrics, some references to consider are as follows:

Two widely used textbooks that discuss many of the procedures in *LIMDEP* are

- Greene, W., *Econometric Analysis*, 7th Edition, Prentice Hall, 2012.
- Wooldridge, J., *Econometric Analysis of Cross Section and Panel Data*, 2nd Edition, MIT Press, 2011.

On the subject of limited and qualitative dependent variables, some useful sources are:

- Maddala, G. S., *Limited Dependent and Qualitative Variables in Econometrics*, Cambridge University Press, 1983.
- Long, S., *Regression Models for Categorical and Limited Dependent Variables*, Sage, 1997.

- DeMaris, A., *Regression with Social Data: Modeling Continuous and Limited Response Variables*, John Wiley and Sons, 2004.

- Greene, W. and Hensher, D., *Modeling Ordered Choices*, Cambridge University Press, 2010.

More generally, on the subjects of microeconometrics, we recommend:

- Cameron, C. and Trivedi, P., *Microeconometrics: Methods and Applications*, Cambridge University Press, 2005.

Two specialized works on count data which are particularly rich in detail and variety are:

- Cameron, C. and Trivedi, P., *Regression Analysis of Count Data*, Cambridge University Press, 1998.

- Winkelmann, R., *Econometric Analysis of Count Data*, 5th Edition, Springer Verlag, 2008.

A useful theoretical volume and an applications oriented survey on stochastic frontier estimation are

- Kumbhakar, S. and Lovell, K., *Stochastic Frontier Analysis*, Cambridge University Press, 2000.

- Greene, W., 'The Econometric Approach to Efficiency Analysis,' Chapter 2 in Fried, H., Lovell, K. and Schmidt, S. (eds.), *The Measurement of Efficiency*, Oxford University Press, 2008.

For those using *LIMDEP* and *NLOGIT* for discrete choice modeling, the primer

- Hensher, D., Rose, J. and Greene, W., *Applied Choice Analysis*, 2nd Edition, Cambridge University Press, 2015

is specifically devoted to techniques provided by *NLOGIT* and develops many applications using *LIMDEP* and *NLOGIT*.

There are numerous survey articles on some of the other topics relevant to *LIMDEP*, particularly in the *Journal of Econometrics*, the *Journal of Applied Econometrics*, and *Foundations and Trends in Econometrics*. Rather than assemble them here, we shall note the relevant sources in the chapters on the models to which they apply.

R1.3 Summary of What's New in Version 11

Version 11 has been in development for four years. The new features include major extensions of the way the program operates and many new models. Previous users will find the following:

Major new features embedded into all estimation and analysis areas, including:

- **Interaction terms** are becoming much more common in empirical models. Every model that you can fit with *LIMDEP* that is defined by a list of variables can include any number of interaction and nonlinear terms, such as *age***educ*, Log(*income*), and *female***educ* + *female***educ*^2. Categorical variables can also be expanded in line in the model instruction rather than being created permanently in the data set.

- **Partial effects** are an essential post estimation step in model development. With our new **PARTIAL EFFECTS** program, you can compute appropriate (average) partial effects for any variable in any model regardless of how complex. Interaction terms and nonlinear functions of variables are all handled by the program. Complex models that involve direct and indirect effects are easily handled as well. Partial effects and models can be simulated at numerous settings of several variables to produce multiple plots of partial effects (with confidence intervals) – again, in any model that you can specify.

- **Panel data** models are provided for nearly all frameworks supported by *LIMDEP*. We have streamlined the handling of panel data with a single setup (declaration) command that automates setting up the appropriate sample for a panel data analysis. Several new panel data models have been added to the program. You can now develop new applications for MAXIMIZE with panel data.

- **Multiple hypothesis tests** can now be built into every model command. We have also updated the command syntax to simplify specifying hypotheses.

- **The WALD command for computing standard errors** will now compute the 'average nonlinear function' and an appropriate standard error for any function that you specify in the command. It will also retain in the data set function values and estimated standard errors for each observation in the sample. Either the delta method or Krinsky and Robb's method may be used. Options for saving and analyzing the results from WALD have been added.

- **Bootstrapping of standard errors and confidence intervals** can be carried out with any model and any statistic (scalar or vector) that you compute with any part of the program.

Many new built-in models and analysis frameworks, including:

- Numerous new forms of the stochastic frontier model,
- Loglinear models such as generalized beta,
- Count data models and several new forms of ordered choice and binary choice models,
- New features added to **MAXIMIZE** for programs that estimate user defined models.

Streamlined appearance of output throughout the program:

In some cases, results have been reduced. In general output has been reformatted to improve readability. New tools are provided for displaying tables of your own results such as estimates from a model that you program with **MATRIX** or **MAXIMIZE**.

R1.4 Documentation

This manual is arranged so that the functions you are most likely to use are the ones you will find documented first. First time users should take the time to read the first three chapters and skim the first few paragraphs of subsequent chapters before beginning serious use. The tutorial contains a few examples which will get you started. On the basis of these, you will be able to do a considerable amount of analysis using *LIMDEP*.

The two parts of this manual are as follows:

LIMDEP Reference Guide

The *LIMDEP Reference Guide* provides program usage, basic econometric methods, such as estimation techniques and how to test hypotheses, and technical material on program functions. Chapter and section numbers in the *LIMDEP Reference Guide* are preceded by the letter ‘R.’

Econometric Modeling Guide

The *Econometric Modeling Guide* describes specific modeling frameworks, such as linear regression, binary choice, stochastic frontier models and survival models. Chapter and section numbers in the *Econometric Modeling Guide* are preceded by the letter ‘E.’

R2: Basics of Operation

R2.1 Introduction to the *LIMDEP* Desktop

Start *LIMDEP* as you would any other program, for example from the *LIMDEP* icon on your desktop. The *LIMDEP* desktop is shown in Figure R2.1.The open window is the project window. The project window contains a listing of the data you will analyze (the variables), results of your analyses (matrices, etc.) and procedures you have used. Right now, you don't have any data in your work area, so the project is empty. This is where you will begin your *LIMDEP* session.

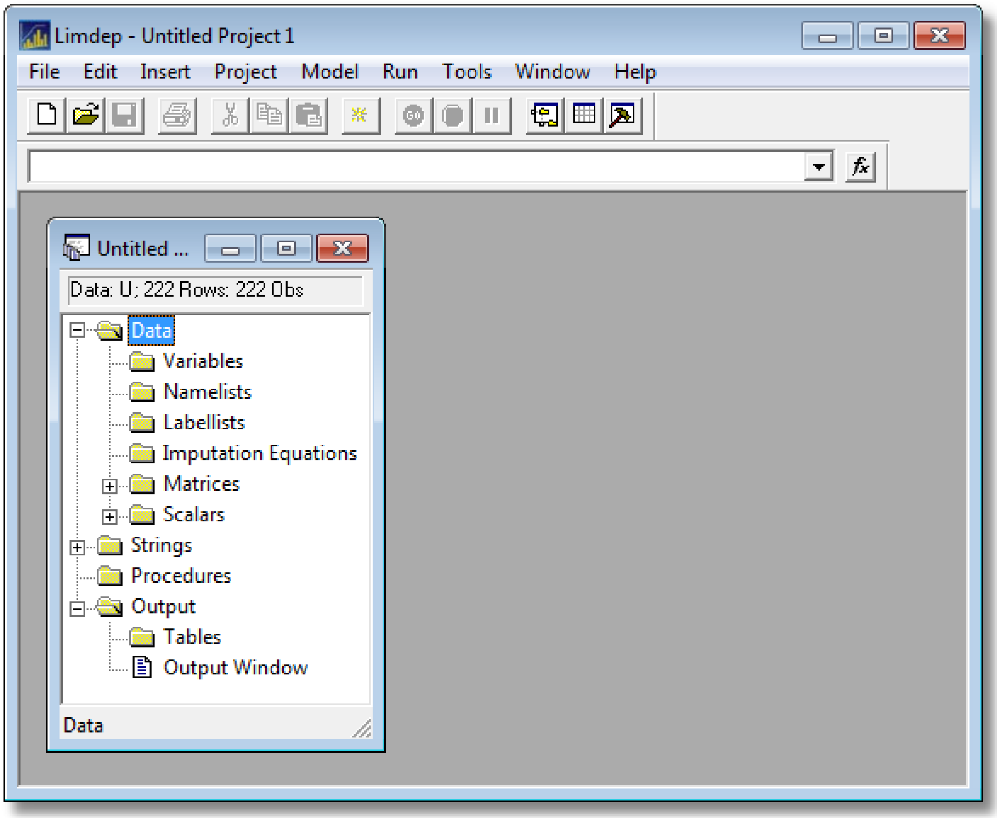


Figure R2.1 *LIMDEP* Desktop Window

R2.1.1 *LIMDEP* Desktop Menus

LIMDEP is operated by menus and dialog boxes as well as by typed instructions (program commands) that you will compose. The menus will mainly be used for management functions such as reading a data set into the program from a file. The program commands will be used for data manipulation such as computing statistics or running a regression.

The menus are at the top of the desktop window. The Project menu and Project:New secondary menu are shown in Figure R2.2. The Project menu is used for reading or writing data and some other operations related to setting up your data set. The menus are described in detail in Section R2.13.2.

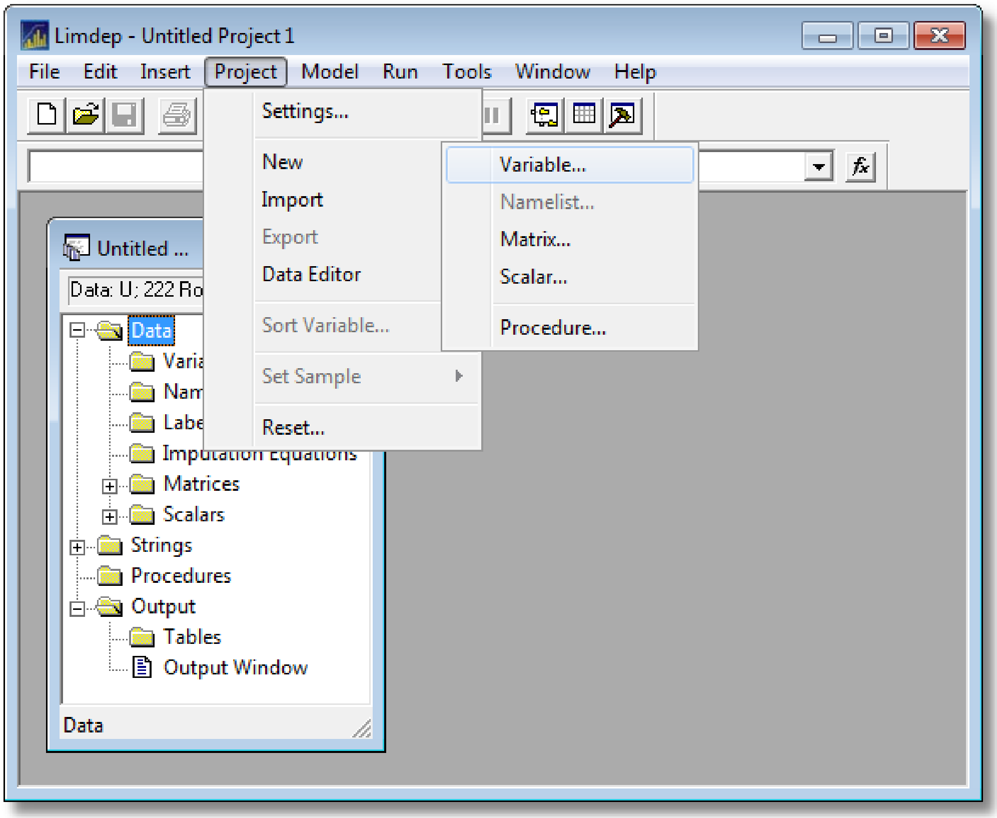


Figure R2.2 The *LIMDEP* Desktop Menus

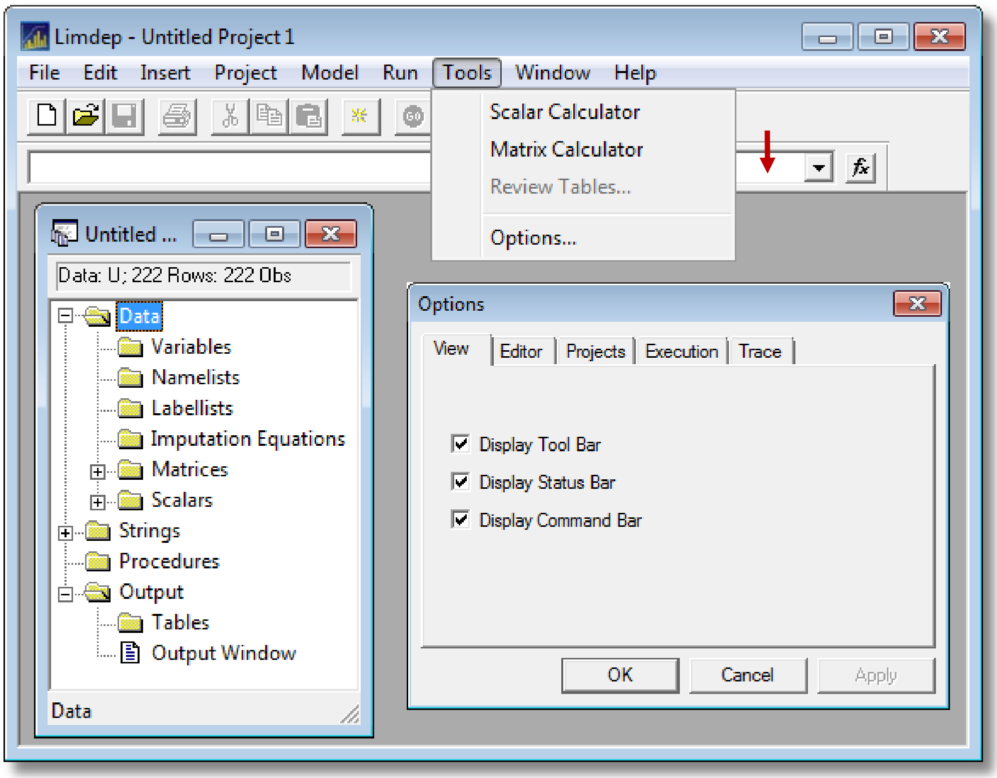
We will frequently use the following shorthand throughout the manual to reference main menu options: Menu Name:Menu Item. For example, the instruction select 'Project:New' indicates select (click) Project from the main menu, then select New from the Project menu. If there are additional options such as in a secondary menu or dialog box, they will be indicated with a forward slash after each option. For example, select 'File:New/Variables'indicatesselect Project from the main menu, select New, then select Variables. This will open a dialog box where you can select new variables.

R2.1.2 The *LIMDEP* Toolbar and Command Bar

The desktop also includes the toolbar and the command bar below the toolbar. If the toolbar or command bar is not showing, select Tools:Options, select the View tab, then check Display Tool Bar or Display Command Bar and click OK. See Figure R2.3.

The *LIMDEP* toolbar contains 14 buttons for shortcuts to various program features. The toolbar buttons are equivalent to certain menu entries. For example, the leftmost opens a new file, the second opens a saved file. Section R2.13.3 contains a description of the toolbar buttons.

The command bar provides a convenient way to submit a short, single line command. It also retains a history of commands submitted from it (like the history kept in the window of a web browser).



[✓] Display Tool Bar
[✓] Display Status Bar
[✓] Display Command Bar

Figure R2.3 The *LIMDEP* Toolbar and Command Bar

R2.1.3 Components of a *LIMDEP* Session

When you are operating *LIMDEP*, you are accumulating a project that consists of at least four components:

- The internal components of the project, including your data, matrices, scalars, the environment, etc. The window associated with this information is the project window.
- The commands that you have accumulated on the screen in an editing window.
- The output that you have accumulated in the output window.
- *LIMDEP*'s session trace that documents the session.

Each of these components will be discussed in more detail throughout the chapter.

R2.2 *LIMDEP* File Types

When you exit, *LIMDEP* will prompt you with a dialog box to ask if you wish to save the contents of the editing, project, and output windows. In each case, you may save the component as a named file.

The project file contains your data. It is *LIMDEP*'s 'save' file and provides a way for you to reenter the program, retrieve your data conveniently and resume your earlier work. The extension for a saved project file is .lpj.

The editing or text window contains the commands that you have accumulated. A saved editing window is referred to as a command or input file. The extension for a saved command file is .lim.

You may also save the contents of the output window. The extension for an output file is also .lim.

WARNING: Output files and command files are both saved with the .lim extension. You will need to make careful note of which files you save are which type.

When you use *LIMDEP*'s dialog box to save the project, editing or output windows, *LIMDEP* will remember the name of the file. When you return, you will be able to select the file from those listed in the File menu. The files listed 1 to 4 are the last four editing or output window files saved by *LIMDEP*, and the files listed 5 to 8 are the last four project files. (See Figure R2.4.) Just click the file name in the File menu to open the file.

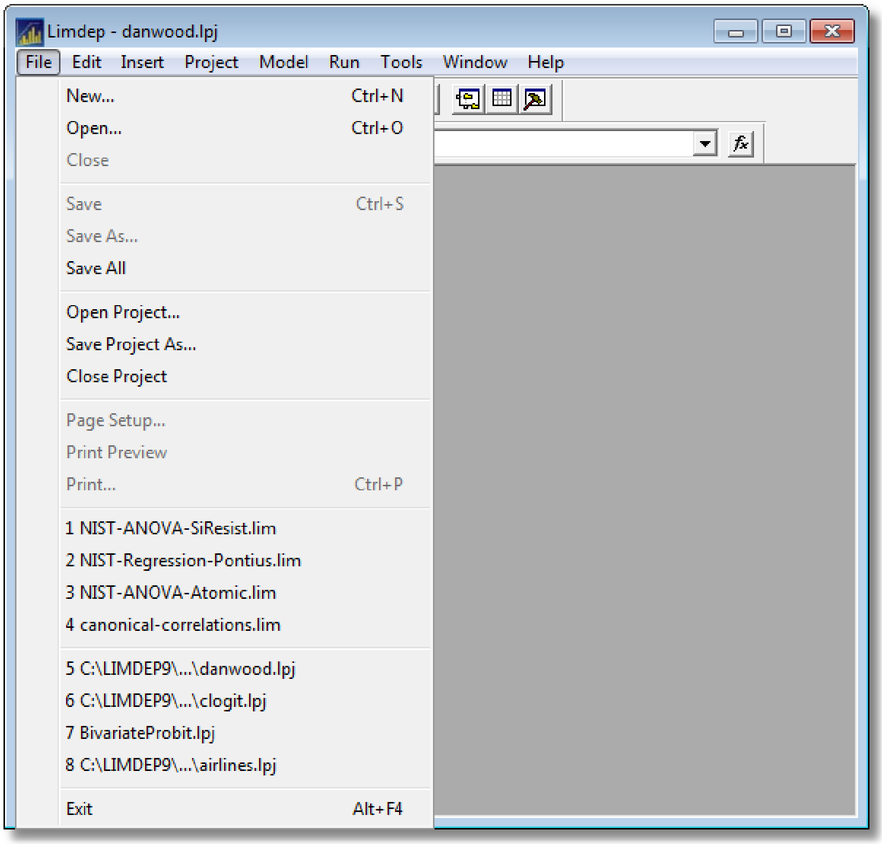


Figure R2.4 The File Menu

During a session, *LIMDEP* accumulates a trace file (trace.lim) that documents the session. The trace file will contain a complete list of your commands exactly as you entered them, all diagnostic messages that were caused by errors in your commands, all diagnostics produced during estimation of models, such as a report of multicollinearity, useful notes about model estimation, such as by what rule an iterative estimator converged. This file is overwritten each time it is created. If you wish to preserve the trace from a session, you should copy it to another file immediately upon leaving your session.

R2.3 Beginning the *LIMDEP* Session

When you begin your *LIMDEP* session, the initial screen will show a project window entitled 'Untitled Project 1' and an empty desktop as shown in Figure R2.1. For a new session in which you intend to analyze a data set that you have not already saved, you should *not* open a new project at this point. The new session already has open project, and you may just proceed to build it. However, at any time during a session, if you wish to open a new project file, you can select File:New/Project/OK. Opening a project file that you have already saved is described in the next section.

R2.3.1 Opening a Project File

There are several ways to retrieve a project file:

- Select File:Open or File:Open Project. This will open a dialog box where you can navigate to the project file you wish to open. Project files have an .lpj ending.
- The four previous projects you opened will be shown as items 5 to 8 in the File menu. You can retrieve any of these files just by clicking the file name in this list, as shown in Figure R2.4.
- You can also open a project and launch *LIMDEP* at the same time. When you double click a file name with the suffix .lpj anywhere on your computer, such as your desktop or an email attachment, Windows will launch *LIMDEP* and then *LIMDEP* will open the project file.

NOTE: In order to operate *LIMDEP*, you must have a project open. This may be the default untitled project or a project that you created earlier. You will know that a project is open by the appearance of a project window on your desktop. Most of *LIMDEP*'s functions will not operate if you do not have a project open.

R2.3.2 Opening an Editing Window

The usual way to submit instructions to *LIMDEP* is by typing program commands (verbs) in an editing window (the text editor or command editor.). To open the editing window, click File:New, select Text/Command Document in the dialog box, and click OK, as shown in Figure R2.5.

TIP: You can press Ctrl-N at any time to bring up the 'New' dialog box.

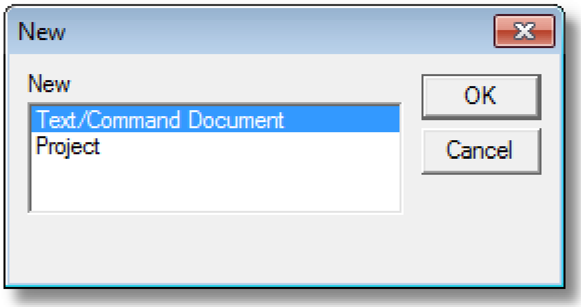
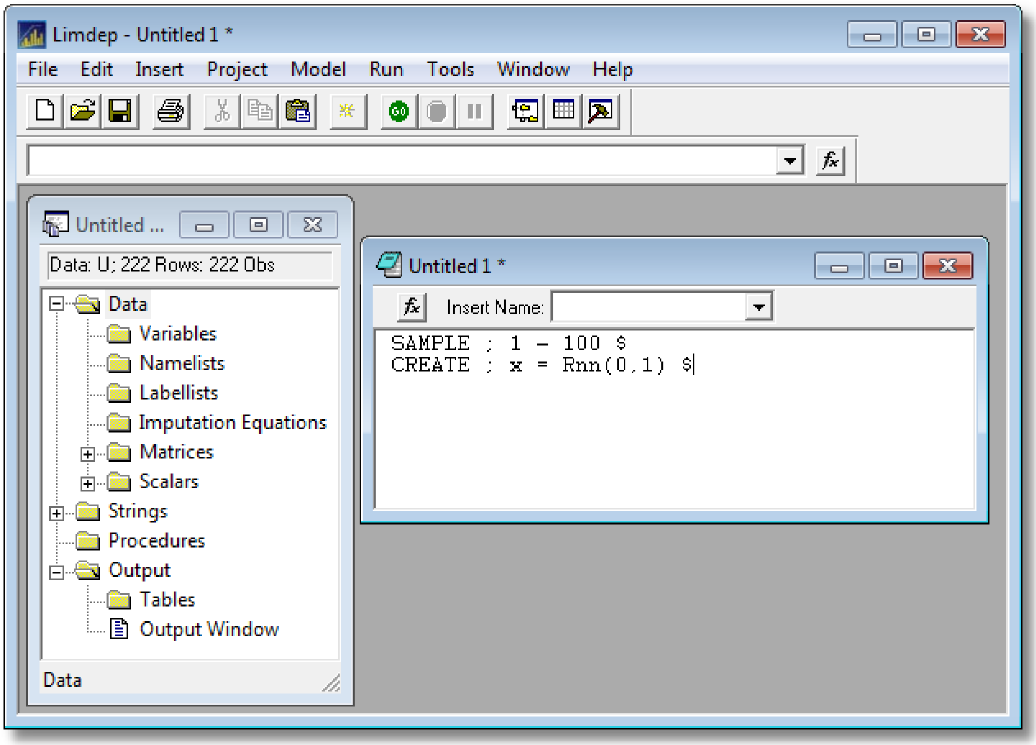


Figure R2.5 The New Dialog Box to Open an Editing Window

The editing window will appear to the right of the project window, as shown in Figure R2.6. You can begin to enter your commands in the editing window as we have done in an example in the figure. The editing window operates as an ordinary text editor, using basic text entry, copy, cut and paste editing features.



```
SAMPLE ; 1 - 100 $
CREATE ; x = Rnn(0,1) $
```

Figure R2.6 Project Window and Editing Window

Note that the editing window shown in Figure R2.6 is labeled 'Untitled 1.'This means that the contents of this window are not associated with a file; the commands in an untitled window are just added to the window during the session. When you open a '.lim' file, the file will be associated with the window, and its name will appear in the window banner. The '*' in the title means that the contents of this window have not yet been saved.

There are other ways to open an editing window:

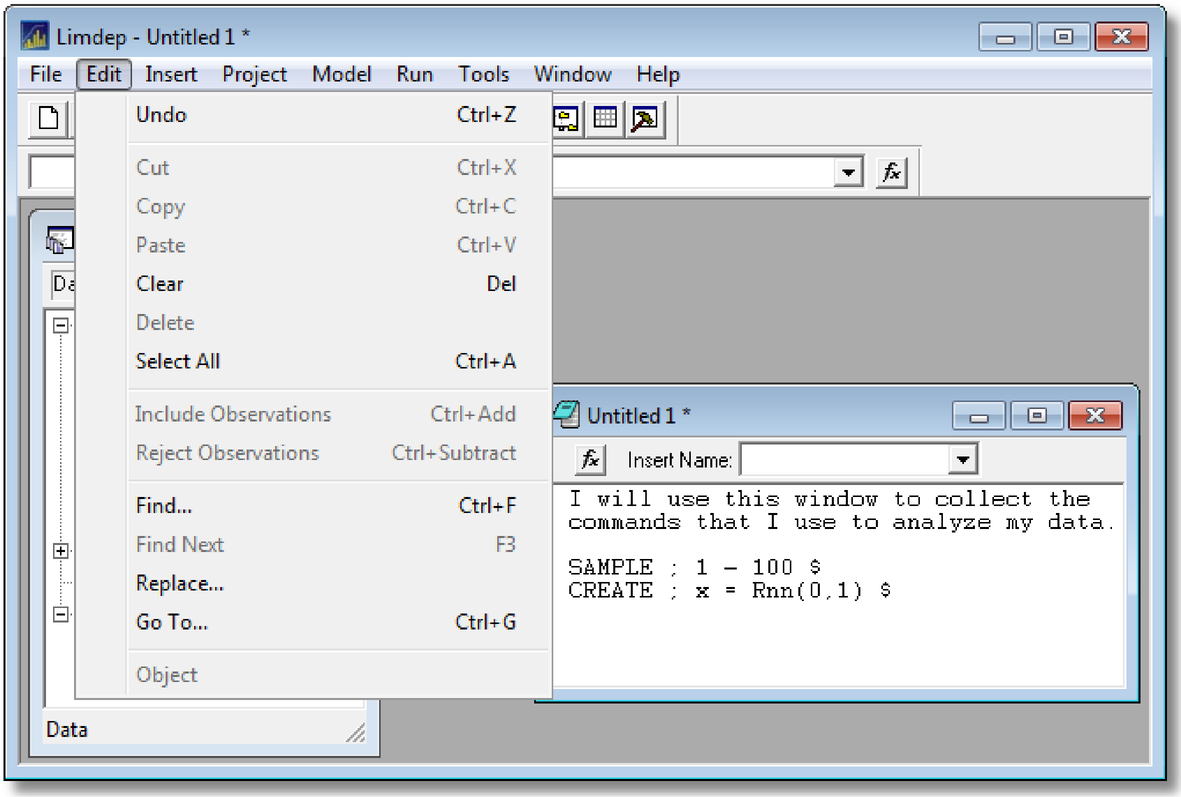
- If you have created a text file (.txt) that contains *LIMDEP* commands you will be using, instead of creating a new set of commands, you can use File:Open to open that file. *LIMDEP* will automatically open an editing window and place the contents of the file in the window.

- You can open an editing window and launch *LIMDEP* at the same time. When you double click a command file name with the suffix .lim anywhere on your computer, such as your desktop, Windows will launch *LIMDEP* and then *LIMDEP* will open an editing window for this command file. Note, however, that when you do this, you must then either open an existing project file or a new project.

R2.4 Using the Editing Window

LIMDEP's editing window is a standard text editor. Enter text as you would in any other Windows based text editor. The Edit menu provides standard editing options such as Undo, Cut, Paste, Copy, Replace, and so on. You can also use the Windows clipboard functions to move text from other programs into this window, or from this window to your other programs. You can, for example, copy text from any word processor, such as Microsoft *Word*, and paste it into the editing window. The *LIMDEP* editing window will inherit all the features in your word processor, including fonts, sizes, boldface, italic, colors, math objects, etc. However, once you save, then retrieve this window, these features will be lost, and all that will remain will be the text characters, in Courier font.

TIP: The text editor uses a Courier, size 9 font. If you are displaying information to an audience or are preparing materials for presentation, you might want to have a larger or different font in this window. You can select the font for the editor by using the Tools:Options/Editor:Choose Font menu. You may then choose a different font and size for your displays. This font will be used in the text editing window, and in the output window.



```
I will use this window to collect the
commands that I use to analyze my data.

SAMPLE ; 1 - 100 $
CREATE ; x = Rnn(0,1) $
```

Figure R2.7 Editing Window and the Edit Menu

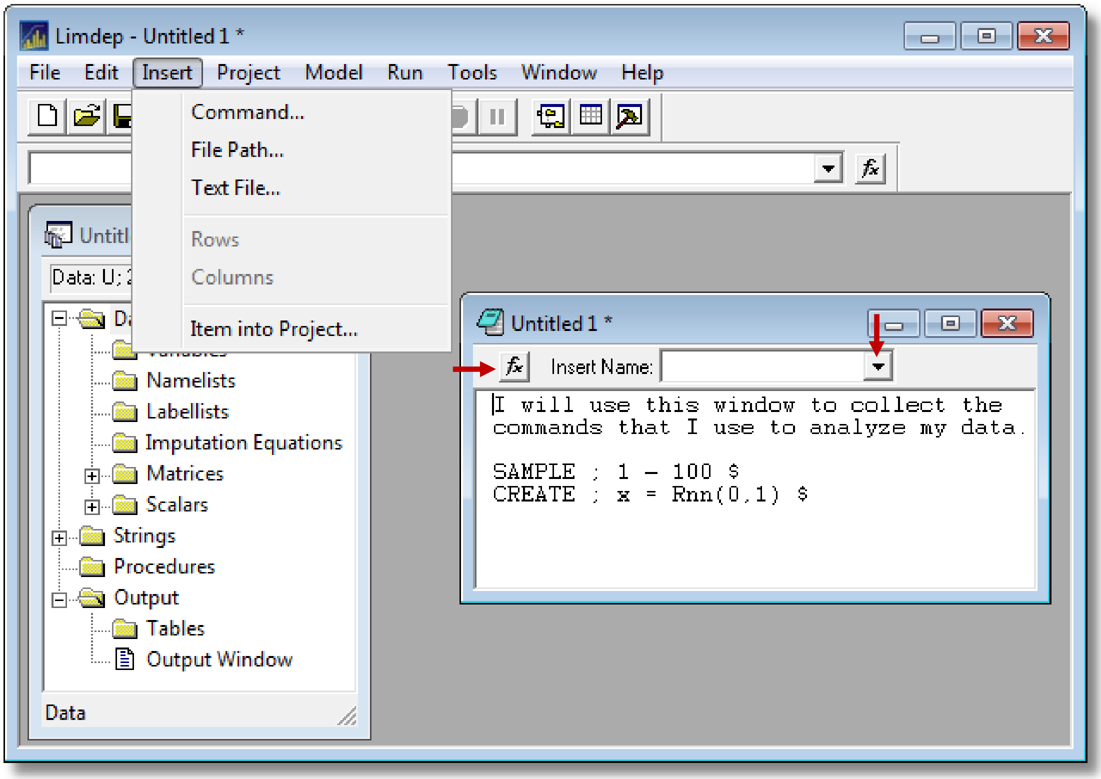
R2.4.1 Using the Insert Menu in the Editing Window

There are additional features that you can use with the editing window. The Insert menu allows you to place specific items on the screen in the editor (see Figure R2.8):

- Insert:Command (or the button marked f_x in the upper left corner of the editing window) will place a specific *LIMDEP* command (verb) at the insertion point. A dialog box will allow you to select the command from a menu or build a model command from a full listing of the options available.

- Insert:File Path will place the full path to a specific file at the insertion point. Several *LIMDEP* commands use files. The dialog box will allow you to find the full path to a file on your disk drive, and insert that path in your command.

- Insert:Text File will place the full contents of any text file you select in the editor at the insertion point. You can merge command files or create command files, using this tool. You can then navigate to, and insert any text file you like.



```
I will use this window to collect the
commands that I use to analyze my data.

SAMPLE ; 1 - 100 $
CREATE ; x = Rnn(0,1) $
```

Figure R2.8 The Insert Menu

TIP: File names must often be enclosed in double quotes for the operating system to find the file that you wish to use. Insert:File Path will include the double quotes when it locates a file name. If you find that *LIMDEP* is unable to find a file that you thought you had specified correctly, make sure that you have included the double quotes.

There are other means to enter names of entities such as variables, matrices, etc. The small ‘Insert Name’ window at the top of the editing window contains a complete list of the names of variables, matrices, etc. that appear in the project window. Click the ▼ button at the right end of the window to see the menu of available names. You can select names from this menu to add to commands as you construct lists in the editing window. You can also drag any name from the project window into the editing window.

R2.4.2 Executing Commands from the Editing Window

When you are ready to execute commands, highlight the ones you wish to submit. Then, to execute the commands you may do either of the following:

- Click GO on the *LIMDEP* toolbar. (If the toolbar is not showing on your screen, select the Tools:Options/View tab, then select Display Tool Bar.)

- Select Run:Run Line (or Run Selection if multiple lines are highlighted) to execute the selected commands once.

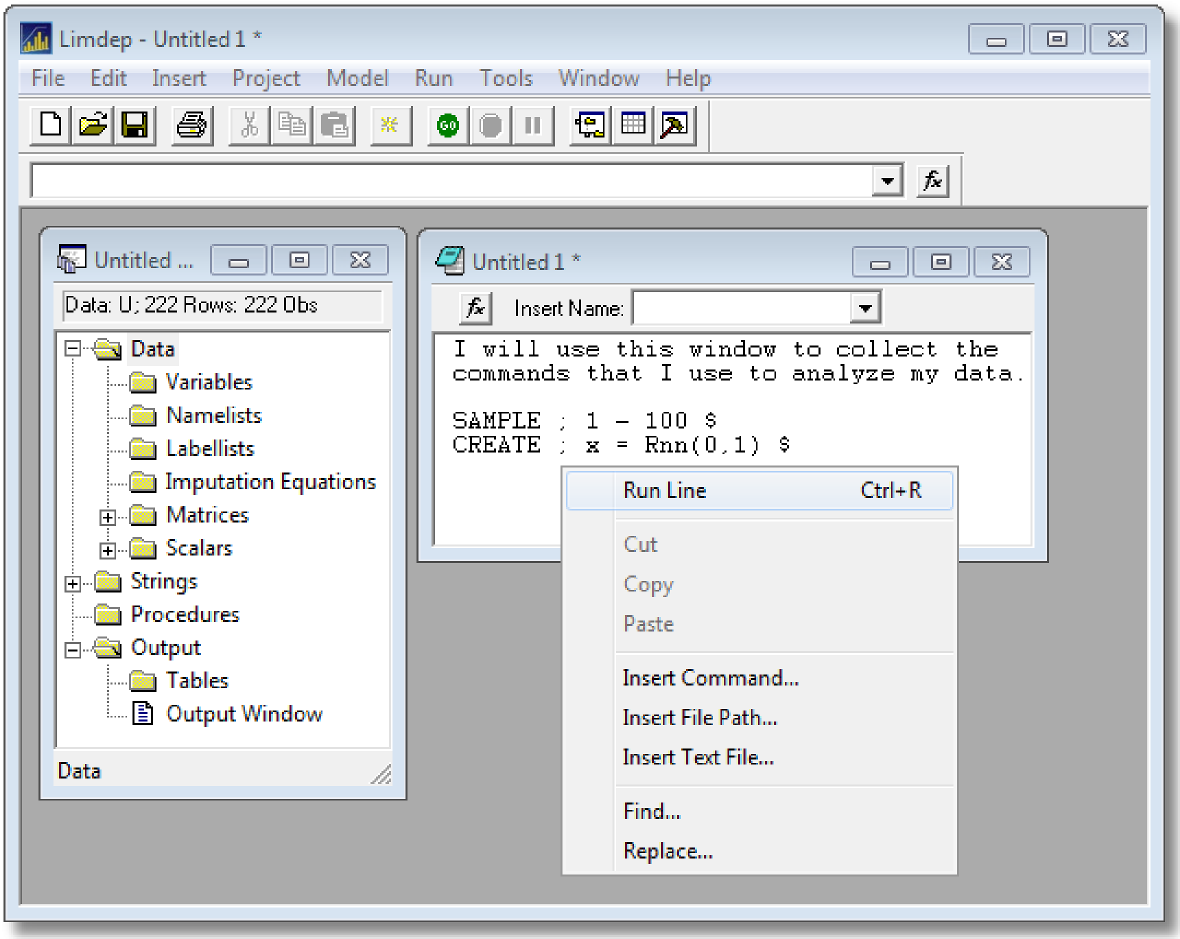
- Select Run:Run Line Multiple Times (or Run Selection Multiple Times if multiple lines are highlighted) to specify that the selected commands are to be executed more than one time. The dialog box queries you for the number of times.

The commands you have selected will now be carried out. In most cases, this will produce some output. *LIMDEP* will now automatically open a third window, your output window, discussed in Section R2.10.

If your commands fit on a single line – many of *LIMDEP*’s commands do not, you can submit a single line of text in editing window just by placing the cursor anywhere on that line (beginning, middle or end), and then clicking the GO button. The single line does not have to be highlighted for this.

R2.4.3 The Editing Window Right Mouse Button Menu

The right mouse button invokes a small menu that combines parts of the Edit and Insert menus, as shown in Figure R2.9. As in the Edit menu, some entries (Cut, Copy) are only active when you have selected text, while Paste is only active if you have placed something on the clipboard with a previous Cut or Copy. Run Line is another option in this menu. Run Line changes to Run Selection when one or more lines are highlighted in the editing window. If you make this selection, those lines will be submitted to the program. If no lines are highlighted, this option is Run Line, for the line which currently contains the cursor.



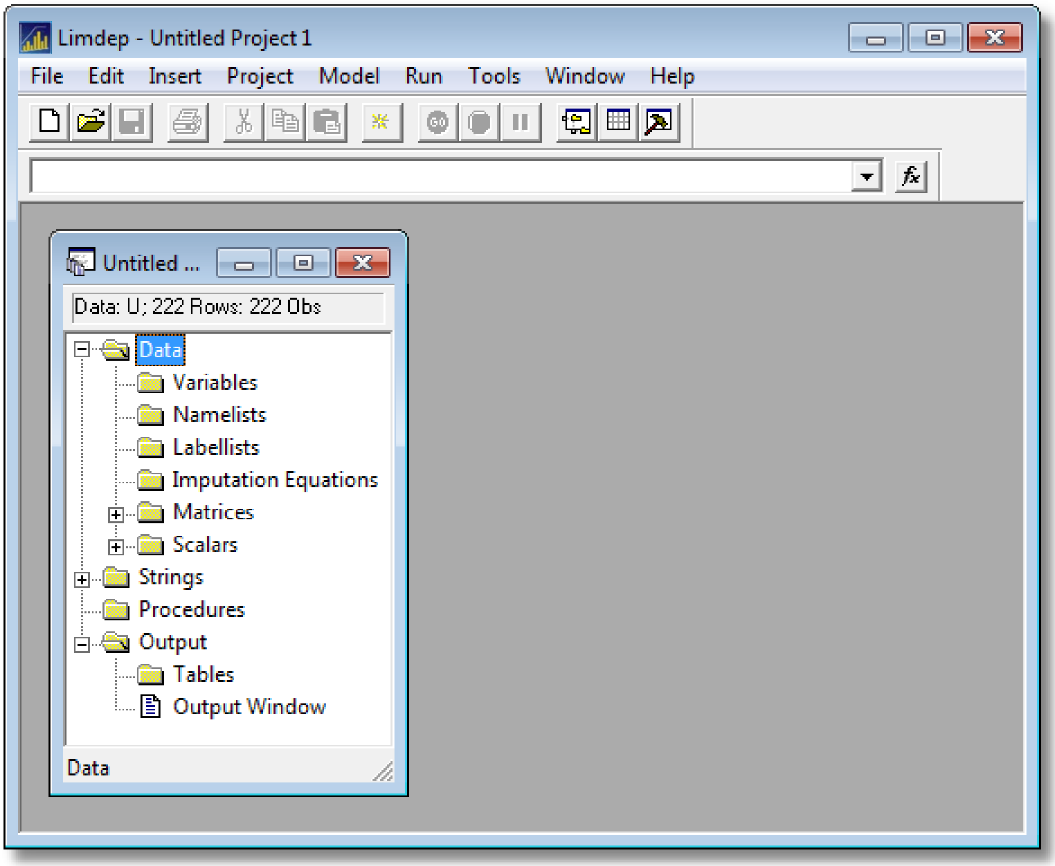
```
I will use this window to collect the
commands that I use to analyze my data.

SAMPLE ; 1 - 100 $
CREATE ; x = Rnn(0,1) $
```

Figure R2.9 Editing Window Right Mouse Menu

R2.5 A Short Tutorial

1. Start the program.

Start *LIMDEP*, for example, by double clicking the shortcut icon on your desktop or from the Start:Programs menu. The desktop will appear as shown in Figure R2.10, with a new project window open, and no other windows active.



Data: U; 222 Rows: 222 Obs

Figure R2.10 Initial *LIMDEP* Desktop

2. Open an editing window.

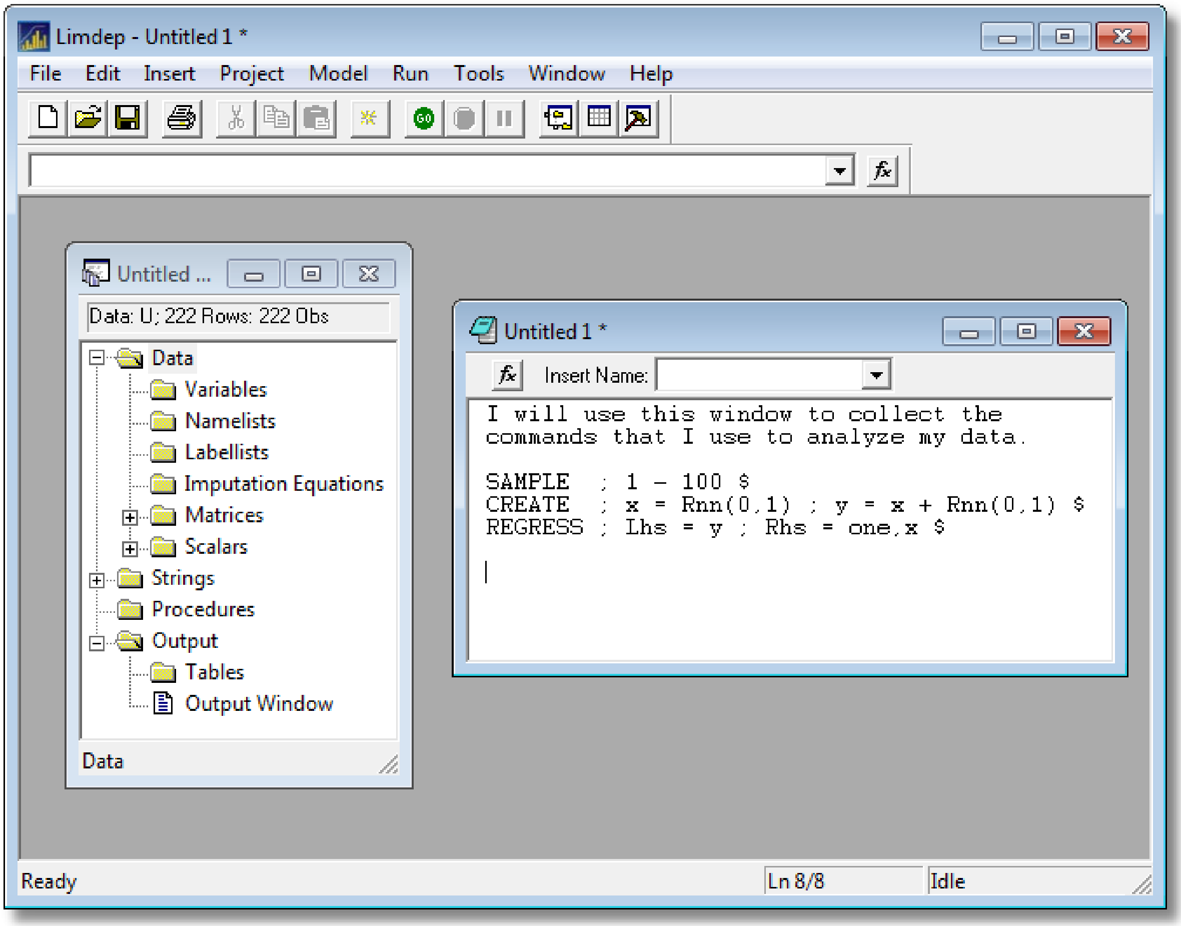
Select File:New, then select Text/Command Document in the dialog box, then click OK, to open an editing window, exactly as discussed in Section R2.3.2.

3. Place commands in the editing window.

Type the commands shown in the editing window of Figure R2.11. These commands will do the following:

1. Instruct *LIMDEP* to base what follows on 100 observations.
2. Create two samples of random draws from the normal distribution, a '*y*' and an '*x*.'
3. Compute the linear regression of *y* on *x*.

Spacing and capitalization do not matter – type these three lines in any manner you find convenient. But, do use three lines.



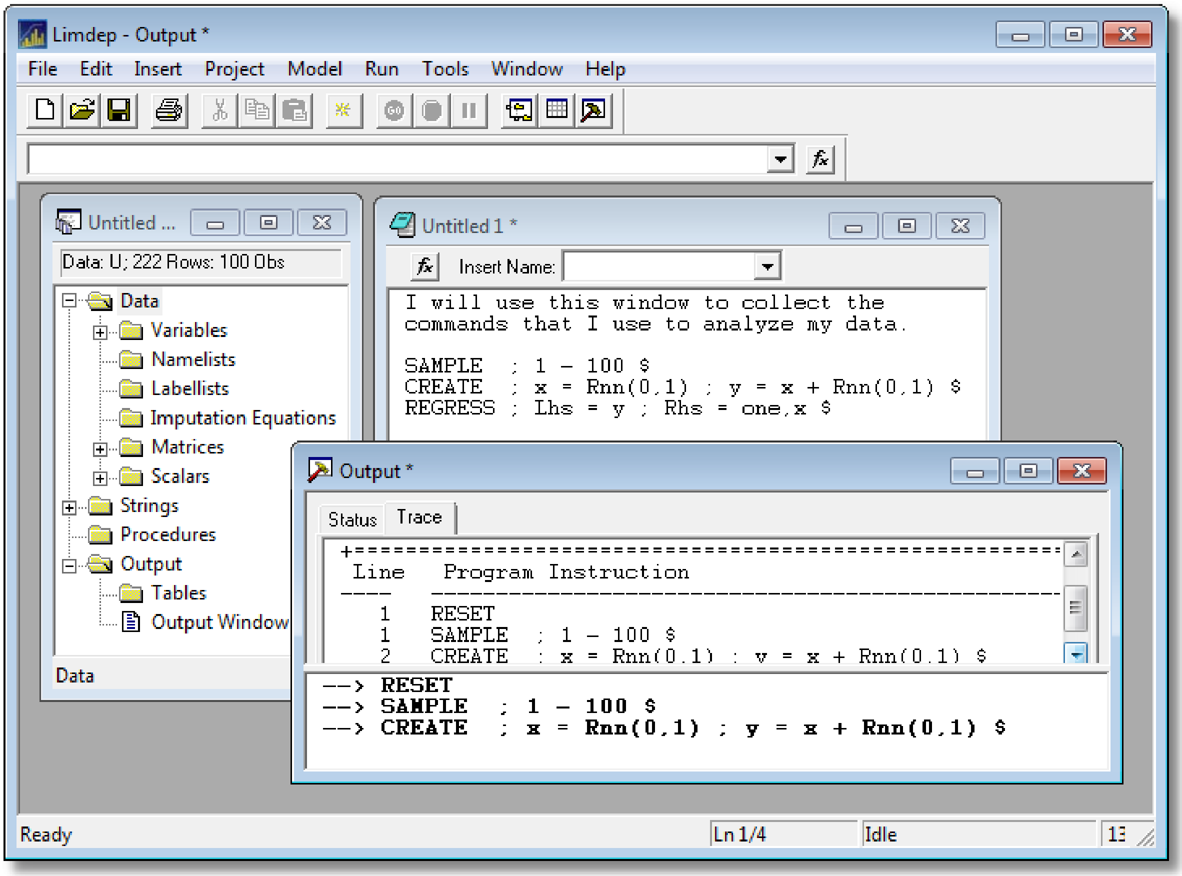
```
I will use this window to collect the
commands that I use to analyze my data.

SAMPLE  ; 1 - 100 $
CREATE  ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
REGRESS ; Lhs = y ; Rhs = one,x $
```

Figure R2.11 Editing Window

4. Submit the first two commands.

Highlight the first two lines of this command set, and click the GO button on the toolbar. Note that a new window appears, your output window, as shown in Figure R2.12. (You may have to resize it to view the output.)

Notice that the top half of the output window has the Trace tab selected. If you click the Status tab, this will change the appearance of the top half of the window, as you'll see later. The status feature in the output window is useful when you execute iterative, complicated nonlinear procedures that involve time consuming calculations. The status window will help you to see how the computation is progressing, and if it is near completion.



```
I will use this window to collect the
commands that I use to analyze my data.

SAMPLE   ; 1 - 100 $
CREATE   ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
REGRESS  ; Lhs = y ; Rhs = one,x $
```

```
+===========================================================
 Line    Program Instruction
 ----    -----------------------------------------------------
    1    RESET
    1    SAMPLE   ; 1 - 100 $
    2    CREATE   ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
```

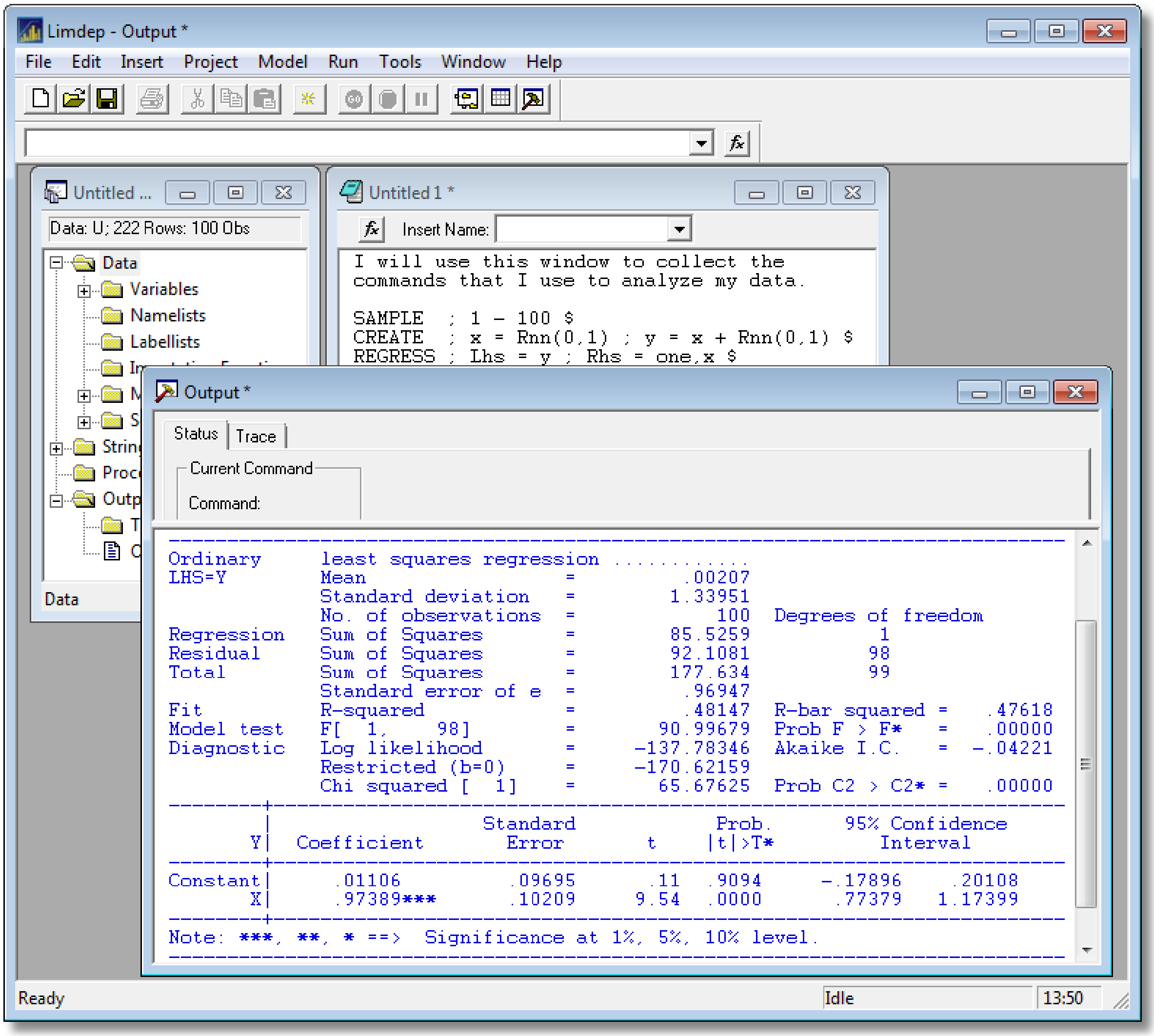
```
--> RESET
--> SAMPLE   ; 1 - 100 $
--> CREATE   ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
```

Figure R2.12 Output Window with Command Echo

The output window will always contain a transcript of your commands. Since you have not generated any numerical results, at this point, that is all it contains.

5. Compute the regression.

Now, select the last line in your command set, the **REGRESS** command, and click the GO button. The regression output appears in the lower half of the window, and you can observe the accumulating trace in the upper half of the window. This trace in the top half of the window will be recorded as the trace file, trace.lim, when you exit the program.



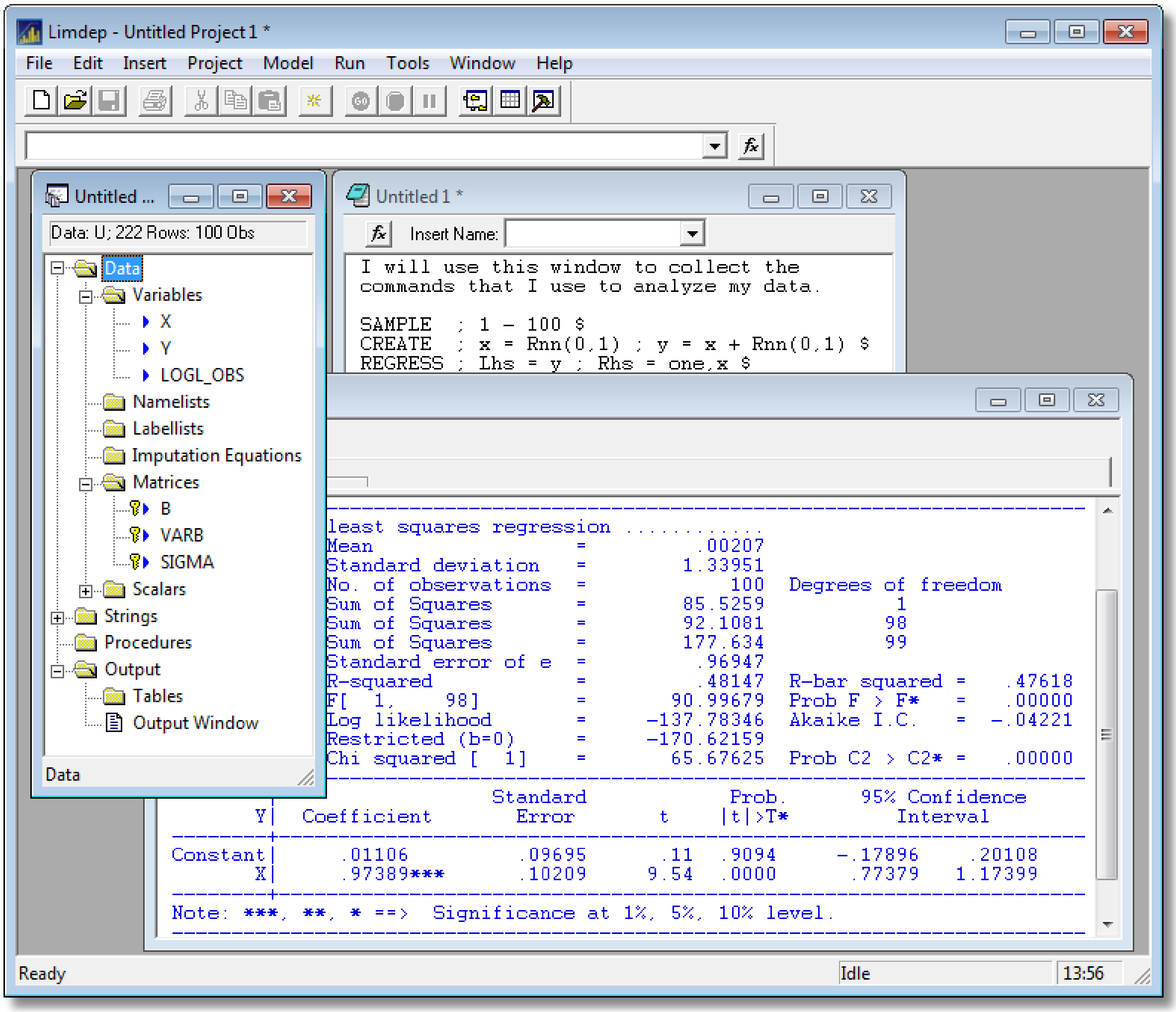
```
I will use this window to collect the
commands that I use to analyze my data.

SAMPLE   ; 1 - 100 $
CREATE   ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
REGRESS  ; Lhs = y ; Rhs = one,x $
```

```
-----------------------------------------------------------------------------
Ordinary    least squares regression ............
LHS=Y       Mean                 =        .00207
            Standard deviation   =       1.33951
            No. of observations  =           100  Degrees of freedom
Regression  Sum of Squares       =       85.5259           1
Residual    Sum of Squares       =       92.1081          98
Total       Sum of Squares       =       177.634          99
            Standard error of e  =        .96947
Fit         R-squared            =        .48147  R-bar squared =   .47618
Model test  F[  1,    98]        =      90.99679  Prob F > F*   =   .00000
Diagnostic  Log likelihood       =    -137.78346  Akaike I.C.   =  -.04221
            Restricted (b=0)     =    -170.62159
            Chi squared [   1]   =      65.67625  Prob C2 > C2* =   .00000
--------+--------------------------------------------------------------------
        |                Standard            Prob.       95% Confidence
       Y|  Coefficient     Error       t    |t|>T*          Interval
--------+--------------------------------------------------------------------
Constant|      .01106       .09695     .11  .9094      -.17896     .20108
       X|      .97389***    .10209    9.54  .0000       .77379    1.17399
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Figure R2.13 Regression Output in Output Window

6. The project window.

Note in Figure R2.10, in the project window, that the topics Matrices and Scalars have ⊞ symbols next to them, indicating that the topic can be ‘expanded’ to display its contents. But, the Variables entry is not marked. After you executed your second line in your editing window, and created the two variables *x* and *y*, the Variables topic is now marked with ⊞. Click this symbol to expand the topic. The **REGRESS** command created another variable, *logl_obs*. It also created three matrices, as can be seen in Figure R2.14. (These three actually exist before you do anything, but they do not contain any values before you fit a model.)



```
I will use this window to collect the
commands that I use to analyze my data.

SAMPLE  ; 1 - 100 $
CREATE  ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
REGRESS ; Lhs = y ; Rhs = one,x $
```

```
-----------------------------------------------------------------------
least squares regression ............
Mean                     =       .00207
Standard deviation       =      1.33951
No. of observations      =          100  Degrees of freedom
Sum of Squares           =      85.5259           1
Sum of Squares           =      92.1081          98
Sum of Squares           =      177.634          99
Standard error of e      =       .96947
R-squared                =       .48147  R-bar squared =    .47618
F[  1,    98]            =     90.99679  Prob F > F*   =    .00000
Log likelihood           =   -137.78346  Akaike I.C.   =  -.04221
Restricted (b=0)         =   -170.62159
Chi squared [  1]        =     65.67625  Prob C2 > C2* =    .00000
-----------------------------------------------------------------------
                        Standard            Prob.      95% Confidence
       Y|  Coefficient     Error       t   |t|>T*         Interval
--------+--------------------------------------------------------------
Constant|      .01106      .09695     .11  .9094     -.17896    .20108
       X|      .97389***   .10209    9.54  .0000      .77379   1.17399
--------+--------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------
```

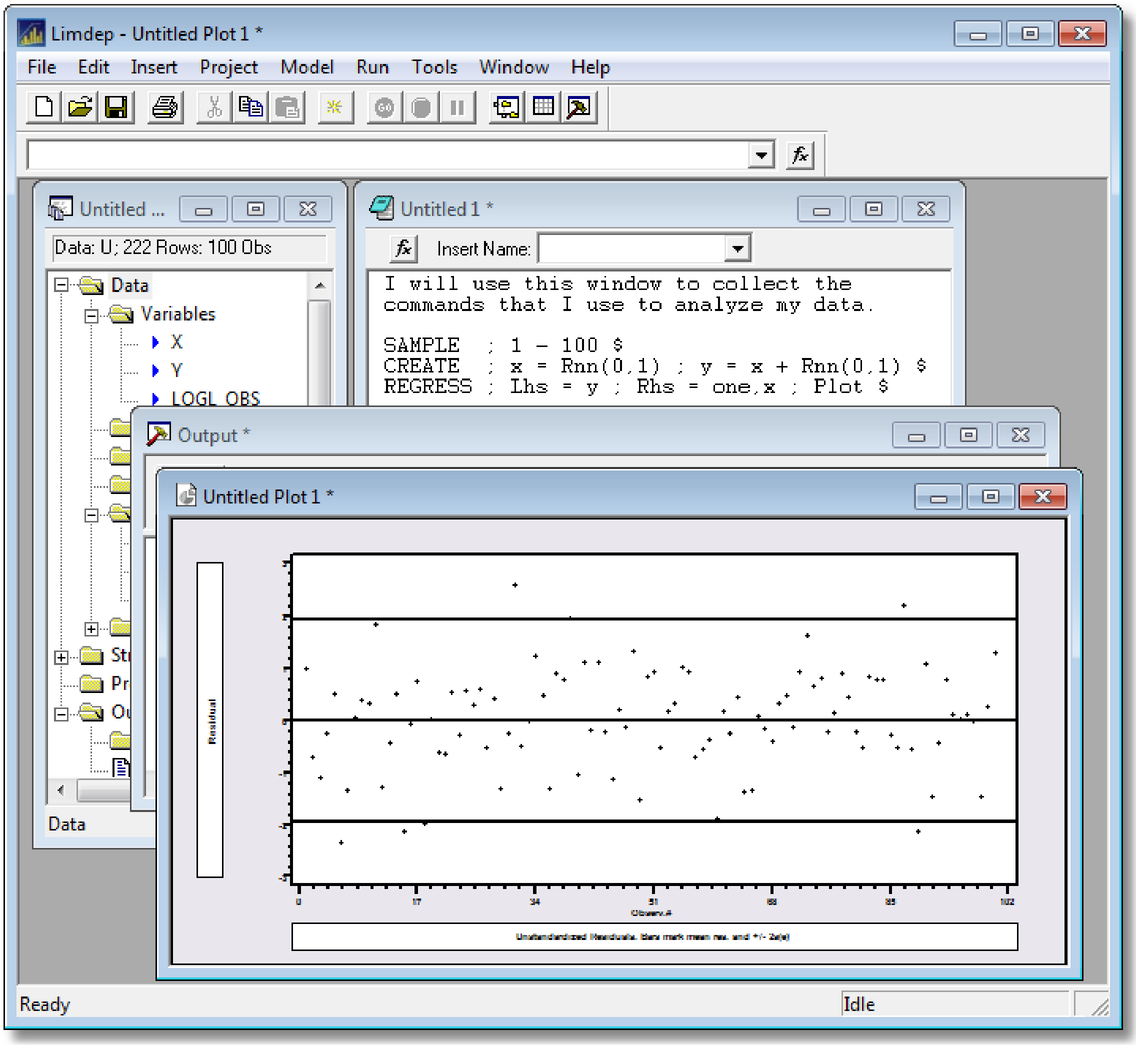
Figure R2.14 Project Window

Some other features you might explore in the project window:

- Click the ⊞ symbol next to the Matrices and/or Scalars topics.
- Double click any name that you find in the project window in any of the three topics.
- Single click any of the matrix or scalar names, and note what appears at the bottom of the window.

7. Modifying commands.

You can return to the editing window and modify the commands and execute them again, in any order. To see an example, move back into the editing window and add **; Plot** to the **REGRESS** command after the ‘*x*’ before the $. After you have changed the **REGRESS** command, resubmit it by clicking the GO button. A new window containing the residual plot you just requested will now appear, as shown in Figure R2.15.


Limdep - Untitled Plot 1 *
File Edit Insert Project Model Run Tools Window Help
Untitled ...
Data: U; 222 Rows: 100 Obs
Data
Variables
X
Y
LOGL_OBS
Data
Untitled 1 *
Insert Name:
Output *
Untitled Plot 1 *
Residual
0
17
34
51
68
85
102
Observ.#
Unstandardized Residuals. Bars mark mean res. and +/- 2s(e)
Ready
Idle

```
I will use this window to collect the
commands that I use to analyze my data.

SAMPLE  ; 1 - 100 $
CREATE  ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
REGRESS ; Lhs = y ; Rhs = one,x ; Plot $
```

Figure R2.15 Plot Window in Output

8. Exiting the program and saving your files.

To leave *LIMDEP*, select File:Exit or simply close the *LIMDEP* desktop window. Whenever you exit a session, you should save your work. At any time in any session, you can save all of *LIMDEP*'s active memory, tables, data matrices, etc. into a file, and retrieve that file later to resume the session. Just select File:Save to save your work during a session.

When you exit, *LIMDEP* will ask if you wish to save the contents of the editing, project, output and other open windows, such as graphs. In each case, you may save the component as a named file. The query in each case is

! Save changes to …<name>…

where <name> is the name that appears in the title banner of each of the active windows. See Figure R2.16 for an example.

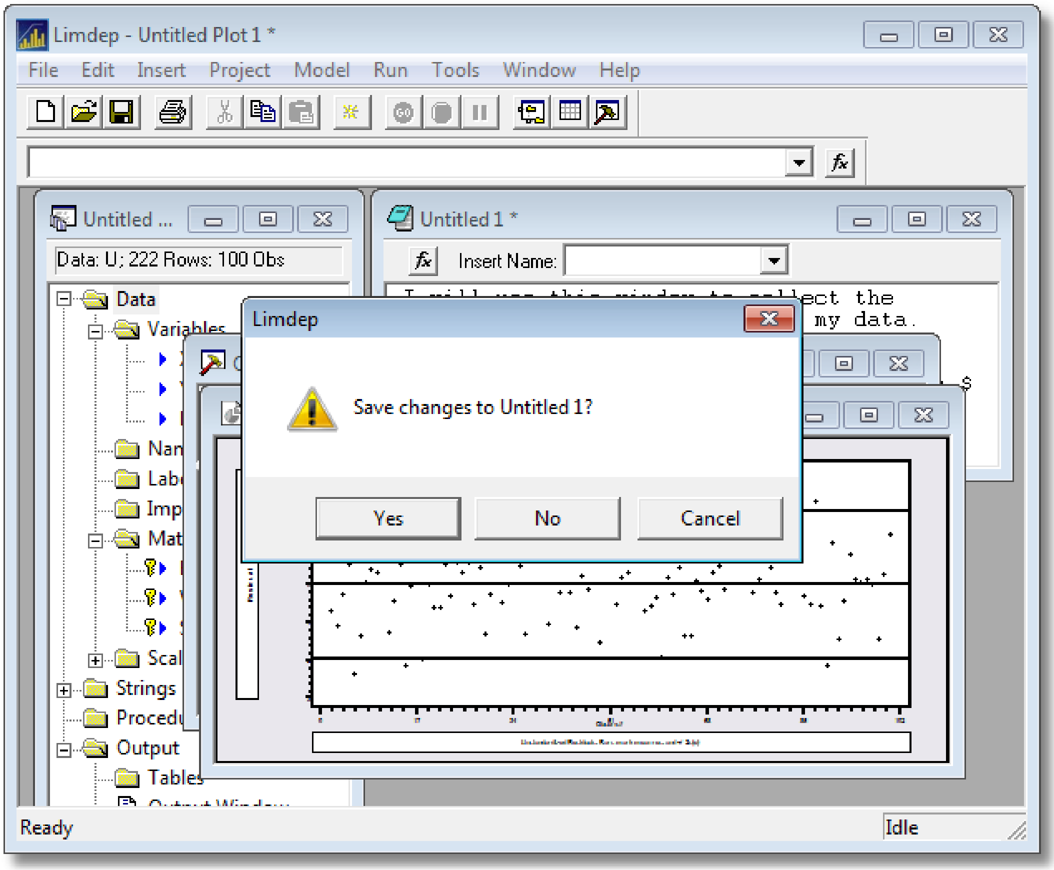

Limdep - Untitled Plot 1 *
File Edit Insert Project Model Run Tools Window Help
Untitled ...
Data: U; 222 Rows: 100 Obs
Data
Variables
Strings
Output
Tables
Untitled 1 *
Insert Name:
Limdep
Save changes to Untitled 1?
Yes
No
Cancel
Ready
Idle

Figure R2.16 Exiting *LIMDEP* – Saving Editor Window Contents

If you click Yes, *LIMDEP* will prompt you for a file name in the Save As dialog box. The extension for a saved project file is .lpj. The extension for a saved editing window command file or output file is .lim. Output files and command files are both saved with the .lim extension. You will need to make careful note of which files you save are which type. For this tutorial, therein need to save any of these windows, so answer no to the four queries about saving your results.

R2.6 Commands

There are numerous menus and dialog boxes provided for giving instructions to *LIMDEP*. (They are described in detail at the end of this chapter.) But, ultimately, the large majority of the instructions you give to the program will be given by commands that you enter in the text editor. This section will describe the *LIMDEP* command language. We begin by describing the general form and characteristics of *LIMDEP* commands.

R2.6.1 Syntax

Commands are of the form:

```
VERB        ; specification ; specification ;  ... ; specification $
```

The verb is a unique four character string which identifies the function you want to perform or the model you wish to fit. If the command requires additional information, the necessary data are given in one or more specifications separated by semicolons (**;**). Commands always end with a dollar sign (**$**). The set of commands in *LIMDEP* consists generally of data setup commands such reading a data file, data manipulation commands such as transforming a variable, programming commands such as matrix manipulation and scientific calculation commands, and model estimation commands. All are structured with this format. Examples of the four groupings noted are:

```
READ        ; File = "C:\work\frontier.dat" ; Nobs = 27 ; Nvar = 4 $
CREATE      ; logq = Log(output) $
MATRIX      ; identity = Iden(5) $
FRONTIER    ; Lhs = logq ; Rhs = one, Log(k), Log(l) ; Model = Exponential $
```

Notice that several of the verbs are more than four characters. Only the first four are strictly necessary, but using the full names helps to document the feature you are using. Thus, **READ** and **READFILE** are the same verb. The following command characteristics apply:

- You may use upper or lower case letters anywhere in any command. All commands are translated to upper case immediately upon being read by the program, so which you use never matters. (Certain labeling and title features for graphs will be exceptions to this.)

- You may put spaces anywhere in any command. (You may also use tabs in an input file.) *LIMDEP* will always ignore all spaces and tabs in any command.

- Every command must begin on a new line.

- The number of nonblank characters which precede the ending **$** must not exceed 10,000.

- In any command, the specifications may always be given in any order. Thus,

  ```
  READ    ; Nobs = 100 ; File = data.prj $ and
  READ    ; File = data.prj ; Nobs = 100 $
  ```

  are exactly the same.

- You may use as many lines as you wish to enter a command. Just press Enter when it is convenient. Blank lines in an input file are also ignored

- Most of your commands will fit on a single line. However, if a command is particularly long, you may break it at any point you want by pressing Enter. The ends of all commands are indicated by a **$**.*LIMDEP* scans each line when it is entered. If the line contains a **$**, the command is assumed to be complete.

HINT: Since commands must generally end with a **$**, if you forget the ending **$** in a command, it will not be carried out. Thus, if you submit a command from the editor and 'nothing happens,' check to see if you have omitted the ending **$** on the command you have submitted. Another problem can arise if you submit more than one command, and one of them does not contain a **$**. The subsequent command will be absorbed into the offending line, almost surely leading to some kind of error message. For example, suppose the illustrative commands we used above were written as follows: Note that the ending **$** is missing from the second command.

```
SAMPLE      ; 1-100 $
CREATE      ; x = Rnn(0,1) ; y = x + Rnn(0,1)
REGRESS     ; Lhs = y ; Rhs = one,x $
```

This command sequence produces a string of errors:

```
Error   623: Check for error in ONE,X
Error   623: Look for: Unknown names, pairs of operators, e.g., *
Error    61: Compilation error in CREATE. See previous diagnostic.
```

The problem is that the **REGRESS** command has become part of the **CREATE** command, and the errors arise because this is now not a valid **CREATE** instruction.

R2.6.2 Naming Conventions and Reserved Names

Most commands refer to entities such as variables, groups of variables, matrices, procedures, and particular scalars by name. Your data are always referenced by variable names. The requirements for names are:

- They must begin with a letter. Remember that *LIMDEP* is *not* case sensitive. Therefore, you can mix upper and lower case in your names at will, but you cannot create different names with different mixes. E.g., GwEn is the same as GWEn, gwen and GWEN.

- You should not use symbols other than the underscore ('_') character and the 26 letters and 10 digits in your names. Other punctuation marks can cause unexpected results if they are not picked up as syntax errors.

- Names may not contain more than eight characters.

There are a few reserved words which you may not use as names for variables, matrices, scalars, namelists, or procedures. These are:

one	(used as a variable name, the constant term in a model),
b, varb, sigma	(used as matrices, to retain estimation results from all models),
n	(always stands for the current sample size),
pi	(the number 3.14159...),
_obsno	(observation number in the current sample, used by **CREATE**),
_rowno	(row number in data set, used by **CREATE**),
s, sy, ybar, degfrdm, kreg, lmda, logl, nreg, rho, rsqrd, ssqrd, sumsqdev	(scalars retained after regressions are estimated),
exitcode	(used to tell you if an estimation procedure was successful).

Several of the reserved names are displayed in the project window. Note in Figure R2.14 that there are 'keys' next to the three matrix names *b*, *varb* and *sigma*. These names are 'locked,' i.e., reserved. You may not change these entities – for example, you may not create a matrix named *b*. That name is reserved for program use.

You are always protected from name conflicts which would arise if you try to give an entity such as a variable a name which is already being used for something else, such as a matrix or scalar, or if you try to use one of the reserved names. For example, you may not name a variable '*s*'; this is reserved for the standard deviation of the residuals from a regression. *LIMDEP* will give you a diagnostic if you try to do so, and decline to carry out the command.

R2.7 Input Files - Entering Commands from a File

Instead of using your text editor, you may submit a set of commands that have previously been placed in a file on your computer. An input (command) file is used to enter commands from a file. Any command may appear in an input file.

There are a few controls that will be useful in an input file. When you use an input file, output such as model results that it produces will come to your screen in normal fashion just as if you had used the editor. (The commands submitted from the editor are, in fact, treated as if they were an input file.)

Normally, you would want to type as little as possible to complete a command. However, for purposes of documenting your commands in an input file or in your trace file, for example, so that you can review a session later, you might want to add commentary to your commands. There are several ways to do so.

Although a verb has a minimum of four characters, you may put any text you like between a verb and the first semicolon or the end of the command if there are no specifications. You may also put comments after the ending **$** in a command. Everything on a line after a **$** is ignored. Thus, to specify a probit model, you might use

```
PROBIT Model ; Lhs = moved ; Rhs = one, age $ Migration model
```

You may also mark parts of your command lines as comments with a question mark (**?**). On any line, any text which follows a **?** is treated as comment and ignored, as is the **?**. For example,

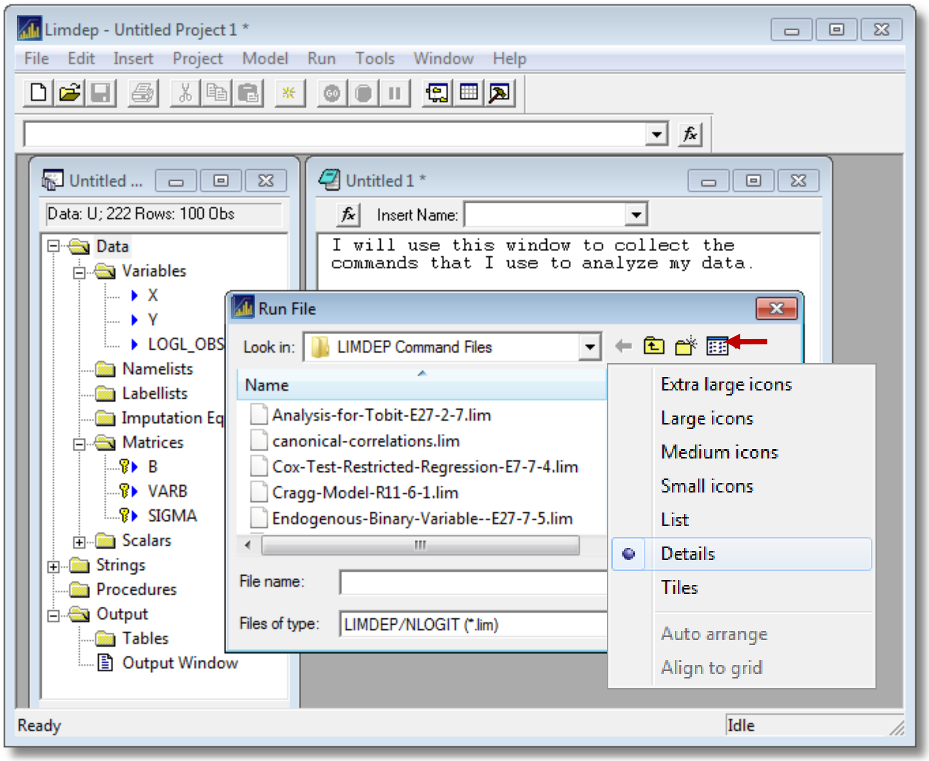
```
LOGIT          ; Lhs = occupatn ? Job choices coded 0,1,2
               ; Rhs = one, age, region $
```

Note, however, that if the **$** appeared at the end of the first line, after the **?**, *LIMDEP* would not find it because it would have appeared as part of a comment. You may also put blank lines anywhere you wish in an input file.

Finally, you may block out a range of lines in an input file as commentary by beginning the first line with '/*' and the last line with '*/.' An example of an input file using these devices follows.

```
/*   This is an example of a LIMDEP input file. The commentary
     is assumed to continue until we end it with a star then a slash.
     (Not yet.)  The first command in this file is going to open a
     file for the program output.                                */
     OPEN  ;  Output  =  demo.out $
 /*  The next line will read a data file.                        */
     READ  ; File  = demo\demo.dat  ? Read is for files on disk
; Nvar  = 3               ? Number of variables
; Nobs  = 50              ? Number of observations
; Names = 1 $              Names at the top of the file
     SAMPLE; 1-50 $
 /*  Compute some transformed  variables. */
     CREATE ; x1x2 = x1*x2 ; x1sq = x1^2 ; x2SQ = x2^2  $
 /*  Now, we fit the 2 regressions. First linear, then loglinear. */
     REGRESS, linear    ; Lhs = y ; Rhs = one,x1,x2,x1sq,x2sq,x1x2  $
     REGRESS, loglinear ; Lhs = Log(y) ; Rhs = one,Log(x1), Log(x2) $
```

To submit an input file as a series of commands to be executed, select Run:Run File to open a dialog box such as the one in Figure R2.17. The file you select is then submitted to *LIMDEP*, as if it were a series of commands that you had submitted from the editor.



I will use this window to collect the
commands that I use to analyze my data.

Analysis-for-Tobit-E27-2-7.lim
canonical-correlations.lim
Cox-Test-Restricted-Regression-E7-7-4.lim
Cragg-Model-R11-6-1.lim
Endogenous-Binary-Variable--E27-7-5.lim

Figure R2.17 Run File Dialog Box

TIP: The dialog box shown in Figure R2.17 is a Windows miniexplorer. You can launch a program, move a file, or delete a file or a folder by operating on the entries in the box. To see the items in the box in details mode rather than in list mode, click the button at the upper right of the dialog box.

R2.8 Work Areas and Projects

When you operate *LIMDEP*, your primary purpose will be to analyze a data set. *LIMDEP* provides a number of 'work areas' in memory. One of them is the 'array' where your data are stored. However, whether you make explicit use of them or not, there are a number of other work areas being maintained for you. It is useful to know about them while you use the program, especially when you approach the limits of their capacity. To summarize, the various entities that you will accumulate and use as you operate *LIMDEP* include:

• Raw data:	The data area.
• Matrices:	Your matrix work area. Your models keep matrix results.
• Scalars:	A bank of named scalars, created by you or by model estimation.
• Namelists:	A set of names that can be used to represent up to 150 other names.
• Procedures:	Possibly large groups of commands that can be submitted at once.
• Imputation Equations:	A set of equations used to impute missing data.
• Tables:	The results of previous models that you have estimated.

In each of these cases, there is a set amount of information that can be stored. You can navigate the project window to find out what entities you have defined and how much room you have left in each of your work areas.

R2.8.1 Work Areas

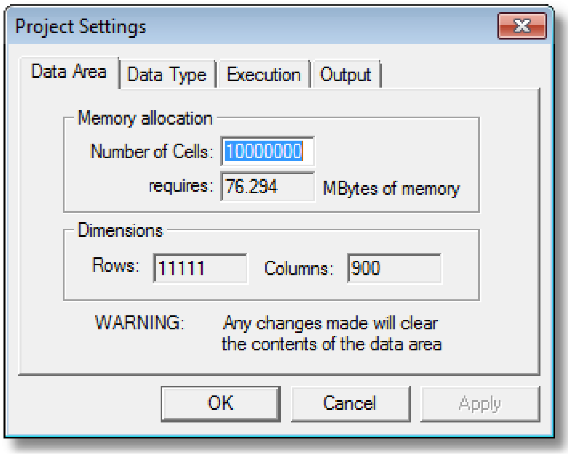
Data Area

The initial setting is 500,000 cells (values) when you start *LIMDEP*. With 900 variables, this allows 555 observations. This is a global setting that you can change if necessary. There are two cases to consider, as shown in Figures R2.18a and R2.18b:

1. To reset the data area size just for the current session, select Project:Settings/Data Area. You can set the dimensions of your data area as needed. In the discussion below, the total size of the data area is referred to as NKMAX. You are not limited by the physical size of the computer, as Windows can swap data from disk to memory as necessary. Note that setting this parameter brings a global program reset. All data are erased. But, this setting is only for the current session.

2. To set the data area size permanently, select Tools:Options/Projects. This sets the default data area size permanently (or until you change it again), so that this will be the setting every time you start *LIMDEP*.

NOTE: In previous versions of *LIMDEP*, if a project (.lpj) file contained an internal data allocation that was larger than the current setting in the program, the **LOAD** command would abort, and the user would be requested to expand the data area before processing could continue. This is now done automatically by the program as part of the **LOAD** operation.


Project Settings

Data Area | Data Type | Execution | Output

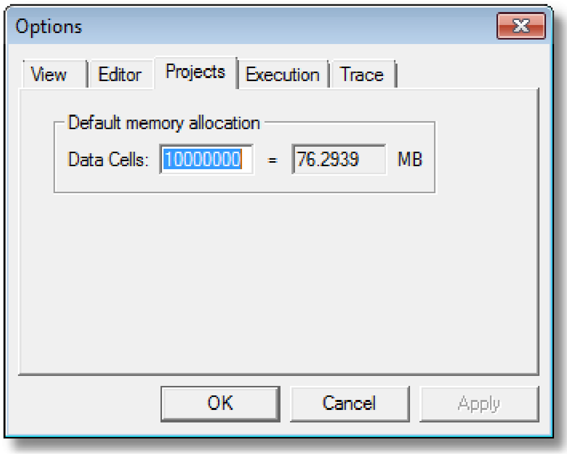
Memory allocation
Number of Cells: 10000000
requires: 76.294 MBytes of memory

Dimensions
Rows: 11111 Columns: 900

WARNING: Any changes made will clear the contents of the data area

OK Cancel Apply

Figure R2.18a Project Settings Data Area


Options

View | Editor | Projects | Execution | Trace

Default memory allocation
Data Cells: 10000000 = 76.2939 MB

OK Cancel Apply

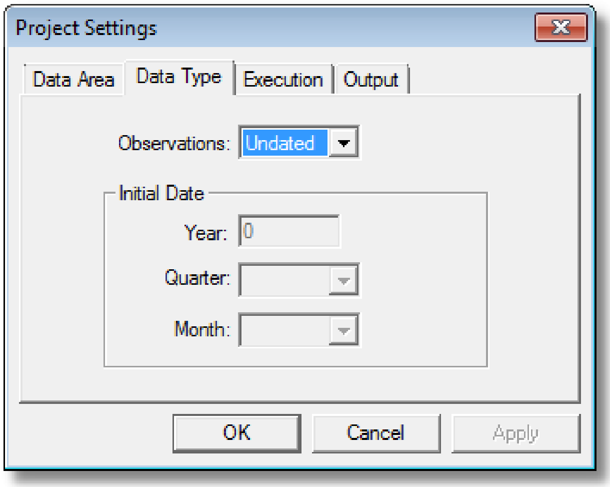
Figure R2.18b Tools Options Default Data Area

Rows and Observations

The number of rows in the data area is the integer part of NKMAX/900. This is only the default. If you need more rows, the adjustment is made at the time you **READ** your data. The number of columns that can be accommodated in the now fixed NKMAX can be adjusted downward if the number of rows is excessive. (900 is a hard upper limit, however.) You can also adjust this setting ‘by hand.’ You would want to do this, for example, if your data were experimental, to be created using a random number generator, and you wanted to analyze more than the default number of observations.

Data Type

This is *Undated* for a cross section. You may specify *monthly*, *quarterly*, or *yearly* for time series data instead. Two ways are to use the **DATA** command or to select Project:Settings/Data Type, and choose the type you wish in the dialog box shown in Figure R2.19. When you choose one of the time series options, only one of the initial date entries is provided.


Project Settings

Data Area | Data Type | Execution | Output

Observations: Undated

Initial Date
Year: 0
Quarter:
Month:

OK Cancel Apply

Figure R2.19 Project Settings Data Type

Variables

You may have up to 900 variables, including *one*, which *LIMDEP* reserves for itself.

Namelists

You may define up to 25 of these, each standing for up to 150 names. See the documentation of the **NAMELIST** command in Section R6.4 for details.

Matrices

You may have up to 100 matrices in a work area that contains 500,000 cells. *LIMDEP* reserves three of these, and 25,000 cells for your model results. This may seem small for possibly large **X** matrices, but, in fact, given the way *LIMDEP* does matrix algebra, you will find it difficult to approach this limit, even if you are manipulating tens of thousands of observations.

Imputation Equations

You may have up to 30 of these stored in a work area. You are unlikely to need more than a small handful; the upper limit should be far more than needed.

Scalars

You may define up to 100 of these, though *LIMDEP* reserves 14 for itself.

Procedures

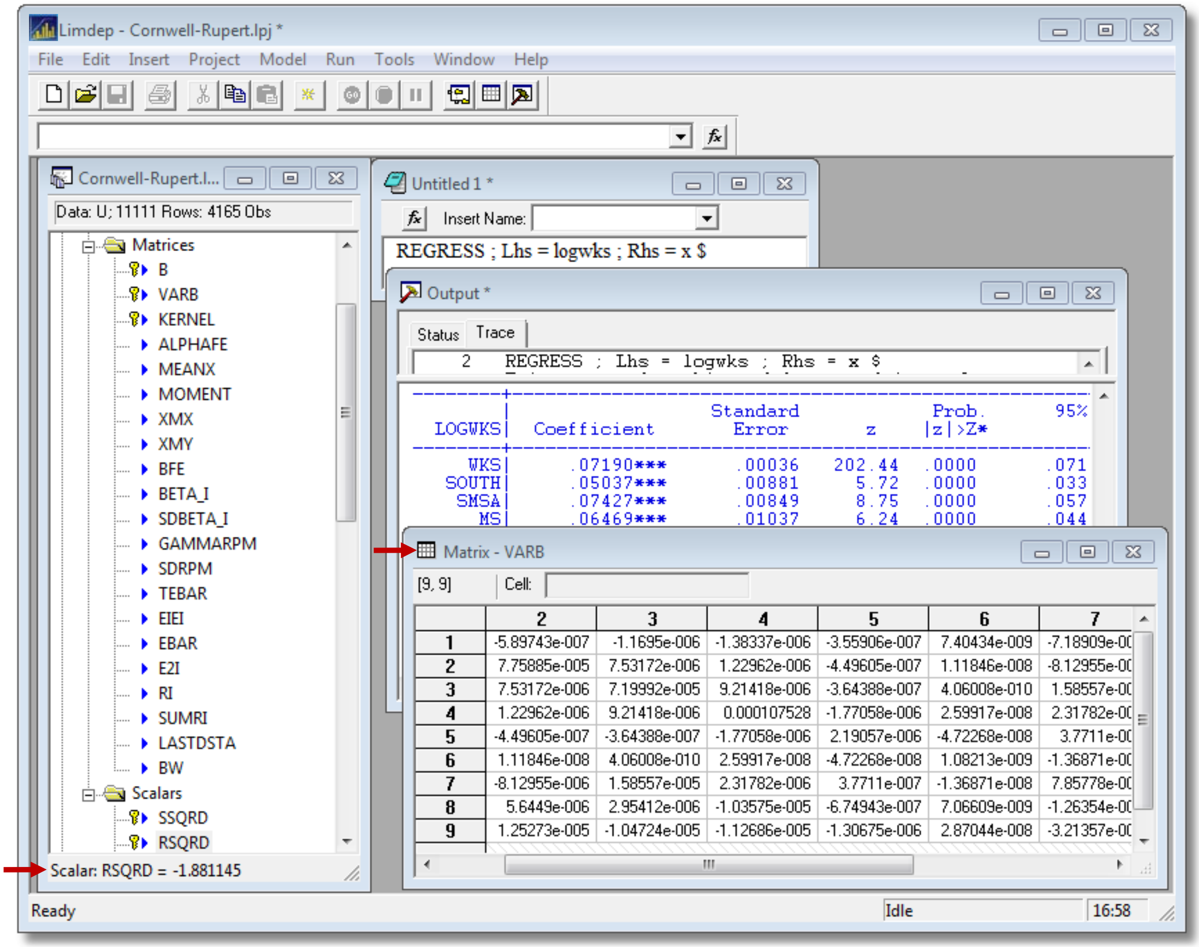
You may define up to 11 of these, 10 in a library and one as the ‘current procedure.’

Tables

You may store for later output to a file the results of 10 models. These may be examined by using the **REVIEW** command.

R2.8.2 The Project Window

Your project window is the leftmost window in Figure R2.14. At any time, you can find an inventory of all of the preceding in the project window. Figure R2.20 shows an example based on the editing window in the figure. There is an inventory of the existing data entities and a display of some of them. Clicking the scalar *rsqrd* displays it at the bottom of the project window. Double clicking the matrix *varb* displays it in a matrix editing window, shown to the right of the project window.



Scalar: RSQRD = -1.881145

```
REGRESS ; Lhs = logwks ; Rhs = x $
```

```
   2   REGRESS ; Lhs = logwks ; Rhs = x $
```

```
--------+------------------------------------------------------------
        |                  Standard            Prob.          95%
  LOGWKS|  Coefficient       Error       z    |z|>Z*
--------+------------------------------------------------------------
     WKS|    .07190***     .00036    202.44   .0000          .071
   SOUTH|    .05037***     .00881      5.72   .0000          .033
    SMSA|    .07427***     .00849      8.75   .0000          .057
      MS|    .06469***     .01037      6.24   .0000          .044
```

[9, 9]

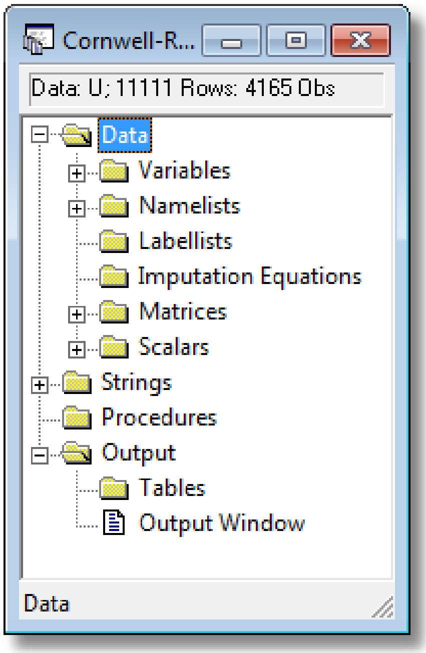
	2	3	4	5	6	7
1	-5.89743e-007	-1.1695e-006	-1.38337e-006	-3.55906e-007	7.40434e-009	-7.18909e-00
2	7.75885e-005	7.53172e-006	1.22962e-006	-4.49605e-007	1.11846e-008	-8.12955e-00
3	7.53172e-006	7.19992e-005	9.21418e-006	-3.64388e-007	4.06008e-010	1.58557e-00
4	1.22962e-006	9.21418e-006	0.000107528	-1.77058e-006	2.59917e-008	2.31782e-00
5	-4.49605e-007	-3.64388e-007	-1.77058e-006	2.19057e-006	-4.72268e-008	3.7711e-00
6	1.11846e-008	4.06008e-010	2.59917e-008	-4.72268e-008	1.08213e-009	-1.36871e-00
7	-8.12955e-006	1.58557e-005	2.31782e-006	3.7711e-007	-1.36871e-008	7.85778e-00
8	5.6449e-006	2.95412e-006	-1.03575e-005	-6.74943e-007	7.06609e-009	-1.26354e-00
9	1.25273e-005	-1.04724e-005	-1.12686e-005	-1.30675e-006	2.87044e-008	-3.21357e-00

Figure R2.20 Project Window and Output Window

There is a tremendous amount of functionality built into the project window. There are four major groupings in the project window, shown in Figure R2.21a Their titles and contents are:

Data:	Variables, namelists, matrices, scalars, labellists, imputation equations
Strings:	A set of three character strings that you can define
Procedures:	Up to 10 named and one unnamed groups of commands
Output:	Output window, model table

Note in Figure R2.21a, that some titles are shown preceded by ⊞, indicating that by double clicking this title or clicking the ⊞, it will be expanded to reveal its contents. Some are shown with a ⊟ to indicate that they are already expanded, and one (procedures) has neither ⊞ nor ⊟ which indicates that there are no procedures to display by expanding this topic. Figure R2.21b shows a more detailed example. Two groupings are expanded in this project window. The following lists the functions available in the project window:



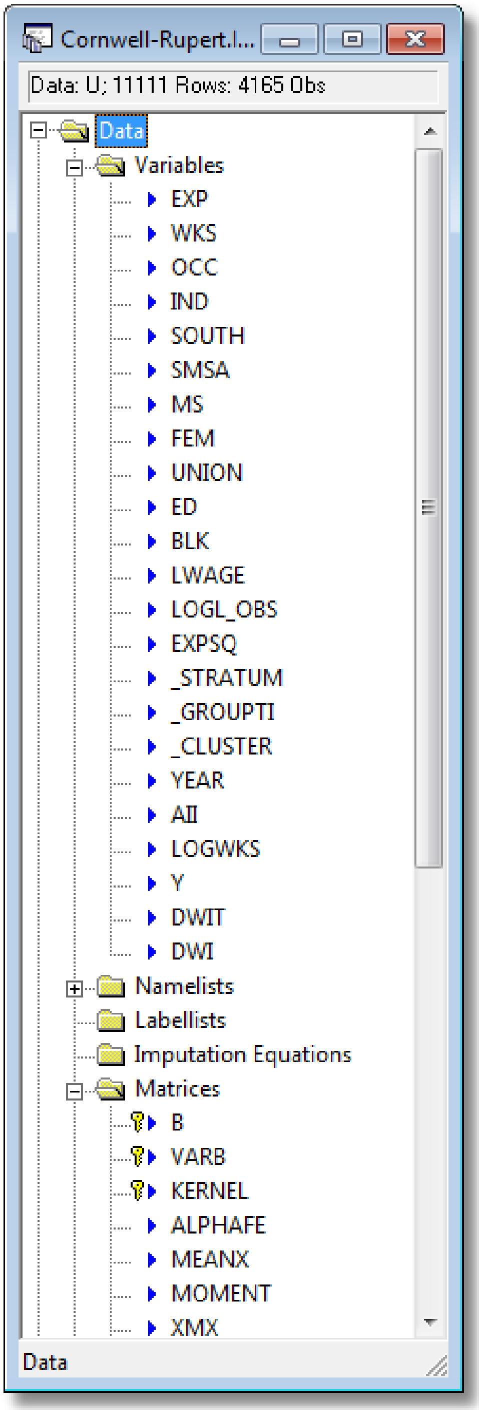


Figure R2.21a Project Window Groupings **Figure R2.21b Expanded Project Window**

Data Group

- Data: Double click the group title to display the six folders in this group.
- Variables: Click the group title to see how many variables currently exist and how many columns there are in your data area.
- Any Variable: Double click any variable name to open the data editor to edit that variable and others that exist at that time.
- Namelists: Click the group title to see how many of the 25 available namelist definitions have been used.
- Any Namelist: Double click any namelist name to enter the namelist editor which will allow you to edit this namelist by adding or deleting variables.
- Labellists: Click the group title to see how many labellists have been defined.
- Any Labellist: Double click the name of a labellist to see the list of labels it defines.
- Imputation Equations: Click the group title to see the names of the variables that can be imputed with the imputation equations.
- Any Imputation Equation: Double click a name in the list to see details of the variables that are used in the imputation equation and what type of equation it is.
- Matrices: Click the group title to see how many of the 100 available matrices have been defined.
- Any Matrix: Click any matrix name to see the dimensions of that matrix displayed at the bottom of the project window.
- Any Matrix: Double click any matrix name to enter an editing window that shows the full matrix and allows you to edit it and save the changes. (*b* and *varb* cannot be changed.)
- Scalars: Click the group title to show how many of your changeable scalars remain available.
- Any Scalar: Click any scalar name, and the value it currently takes will be displayed at the bottom of the project window.
- Any Scalar: Double click any scalar name to enter an editing window which will allow you to replace the value of the scalar. Note, the first 14 scalars are read only.
- Any Variable, Matrix, Scalar, or Namelist: Highlight the name of the entity, then press Del to delete the item from the work area. This may be necessary to clear space.

NOTE: There are various items in the project that are ‘read only.’ These correspond to the reserved names listed earlier. You will know that an item in the project is read only by its identifying key marker, 🔑. This marker indicates an item that you can view in one of the various editors, but cannot change.

Strings Group

- Strings: Double click the group title to open the list of strings.
- Strings: Double click any of the string names to enter an editing window that allows you to define the character string. (See Chapter R19 for use of these strings.)

Procedures Group

- Procedures: Click the group title to see how many of the 10 procedures are available.
- Procedures: Double click the group title (or open the folder) to display the names of the defined procedures.
- Any Procedure: Click a procedure name to display how many lines of commands are in this procedure, of 50 that can be used.
- Any Procedure: Double click any procedure name to open an editing window in which you can edit that procedure and, if you wish, change its parameter list.

Output Group

- Output: Double click the group title to display the two items in the group, tables and output window.
- Tables: Click the group title to display the names (up to 10) of the models that you have stacked in the results table work area. (Nothing happens here if you have not stacked any model results in the table.)
- Any Table Name: Double click any name in the tables grouping to open the editing window where you can construct output tables for model results.
- Output Window: Double click this window entry to activate the output window.

TIP: You can select (highlight) any name in the project window, then drag that name into the editing window if you would like to use the name in constructing commands. Another way to copy a name from the project window into the editing window is to use Ctrl-Click – that is, put the mouse cursor on the name you wish to copy, press and hold down the Ctrl key and click the left button on your mouse. This will allow you to assemble a list of names in the editing window quite quickly. (You will have to add commas to separate the names.)

Other Functions in the Project Window

Many other editing features are built into the project window. By selecting any name in this window with a *right mouse* click, you obtain a menu of features. These are:

- Variable: Data editor, rename the variable, sort the variable, copy the name to the editing window, delete the variable.
- Namelist: Namelist editor, copy the name to the editing window, delete the namelist.
- Matrix: Matrix editor window, copy the name to the editing window, delete the matrix.
- Scalar: Scalar editor/new entry, copy the name to the editing window, delete the scalar. (When in a scalar editing window, right click invokes an editing menu.)
- String: String editor, copy the name to the editing window.
- Procedure: Edit the procedure, run the procedure, copy the name to the editing window.
- Table: Editor, review output tables, add tables to output.

We'll revisit these features at appropriate points later in this manual.

The Insert:Item into Project Menu

The Insert menu includes an option, Item into Project, that offers dialog boxes for creating most of the major data entities that exist in your session, variables, scalars, matrices, namelists, and procedures. Each of these is an editor that allows you to edit an existing entity or to create a new one. See Figure R2.22.

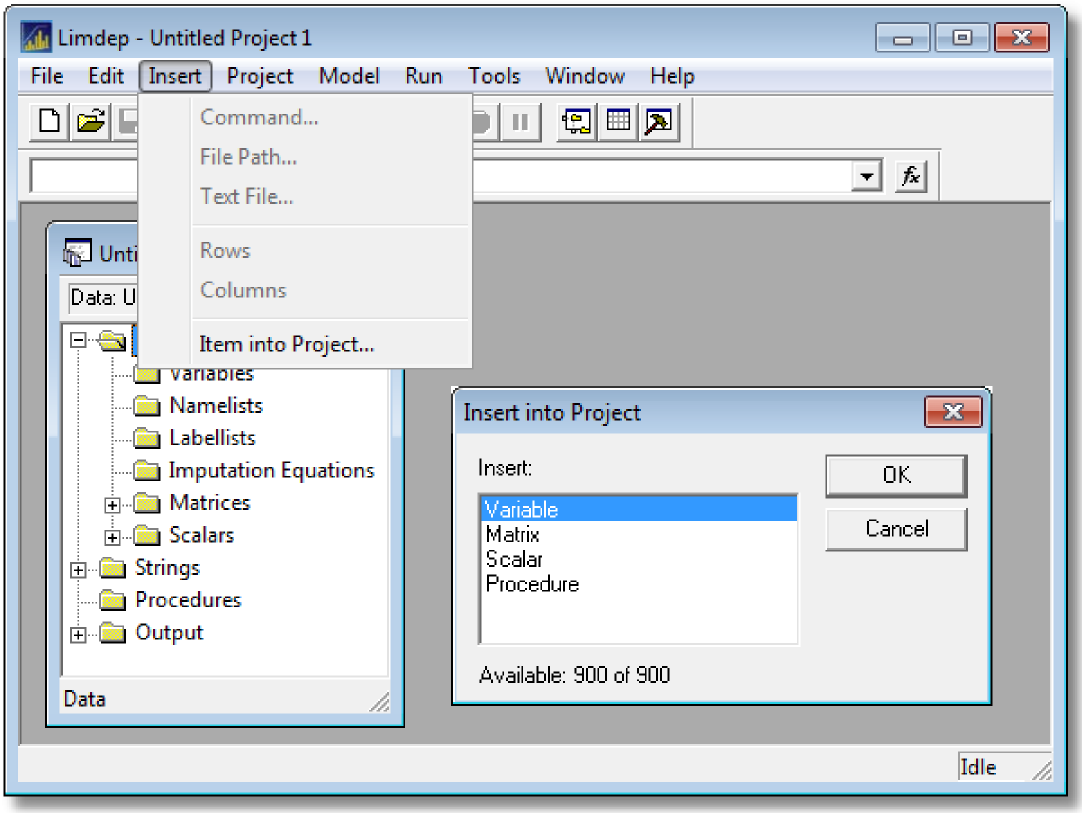


Figure R2.22 Insert Item into Project

R2.9 Restarting During a Session

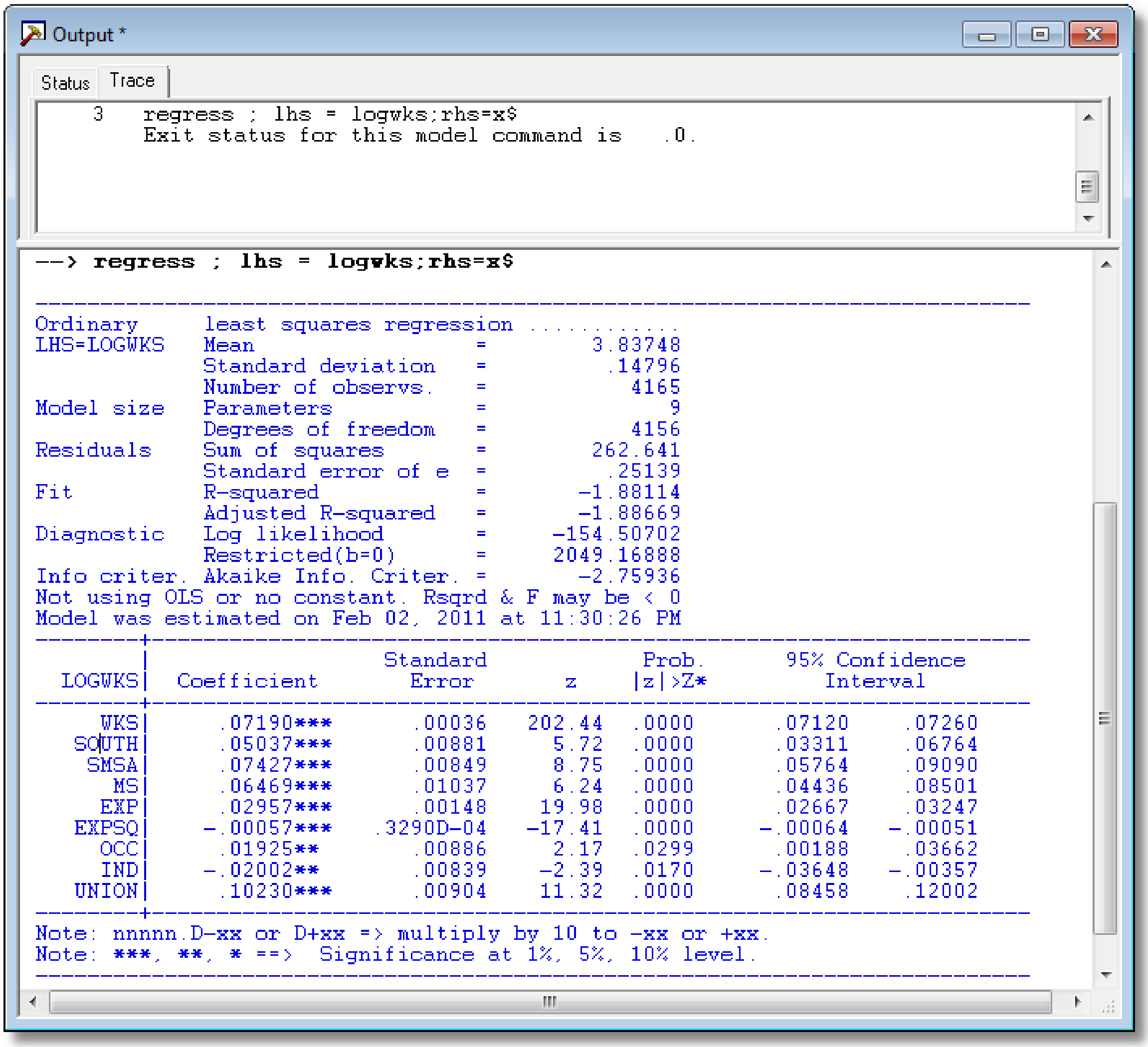
A large amount of information is accumulated during a session. If you wish to begin a new session, with a different data set, for example, it is best to 'sweep' the memory before doing so. The best way is to select File:Exit and restart. It may be more convenient just to clear the memory by selecting Project:Reset (or use the *LIMDEP* **RESET** command). After your confirmation, all memory is cleared and a new session begins.

NOTE: This is a complete reset. All data information is lost. Use File:Saveif necessary, first.

Project:Reset clears all the program memory. But, it does not clear the output window nor does it sweep the text editing window. If you want a completely new session, you should either select File:Exit and restart, or use Edit:Select All then Edit:Clear in each of the two remaining windows.

R2.10 Program Output and the Output Window

LIMDEP will automatically open an output window and use it for the display of results produced by your commands. Figure R2.23 shows an example. The output window is split into two parts. In the lower part, an echo of the commands and the actual statistical results are accumulated. The upper part of the window displays the trace.lim file as it is being accumulated. Note that there are two tabs in the upper window. You have two options for display in this window. The Trace display is as shown below. If you select the Status tab, instead, this window will display technical information during model estimation, such as the iterations, line search, and function value during maximum likelihood estimation, and execution time if you have selected this option from the Project:Project Settings/Execution tab as well. We will review the Status tab in Chapter R26 where we discuss the optimization procedures.



```
      3    regress ; lhs = logwks;rhs=x$
           Exit status for this model command is    .0.
```

```
--> regress ; lhs = logwks;rhs=x$

-----------------------------------------------------------------------------
Ordinary    least squares regression ............
LHS=LOGWKS  Mean                  =         3.83748
            Standard deviation    =          .14796
            Number of observs.    =            4165
Model size  Parameters            =               9
            Degrees of freedom    =            4156
Residuals   Sum of squares        =         262.641
            Standard error of e   =          .25139
Fit         R-squared             =        -1.88114
            Adjusted R-squared    =        -1.88669
Diagnostic  Log likelihood        =      -154.50702
            Restricted(b=0)       =      2049.16888
Info criter. Akaike Info. Criter. =        -2.75936
Not using OLS or no constant. Rsqrd & F may be < 0
Model was estimated on Feb 02, 2011 at 11:30:26 PM
--------+--------------------------------------------------------------------
        |               Standard            Prob.      95% Confidence
  LOGWKS|  Coefficient    Error       z    |z|>Z*        Interval
--------+--------------------------------------------------------------------
     WKS|    .07190***    .00036   202.44  .0000      .07120     .07260
   SOUTH|    .05037***    .00881     5.72  .0000      .03311     .06764
    SMSA|    .07427***    .00849     8.75  .0000      .05764     .09090
      MS|    .06469***    .01037     6.24  .0000      .04436     .08501
     EXP|    .02957***    .00148    19.98  .0000      .02667     .03247
   EXPSQ|   -.00057***  .3290D-04  -17.41  .0000     -.00064    -.00051
     OCC|    .01925**     .00886     2.17  .0299      .00188     .03662
     IND|   -.02002**     .00839    -2.39  .0170     -.03648    -.00357
   UNION|    .10230***    .00904    11.32  .0000      .08458     .12002
--------+--------------------------------------------------------------------
Note: nnnnn.D-xx or D+xx => multiply by 10 to -xx or +xx.
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Figure R2.23 Output Window

R2.10.1 Opening an Output File

You can open an output file if you wish – see Section R9.7. The command is

OPEN ; Output = the desired filename $

All the model results that are sent to your output window will be echoed to this file. One difference is that the file will not contain the interleaved commands, as appear in the example above.

HINT: It is not necessary to open an output file to retain your results *during* your session. As you exit *LIMDEP*, you will be asked if you wish to 'Save changes to <title of output window>?' At this point, if you answer yes, you will be able to create an output file. You will be queried for the name. The file will contain all of the results that have been accumulated during your session.

R2.10.2 Editing Your Output

The output window provides limited capability for editing. You can select, then delete any of the results in the window. You can also highlight, then use cut or copy in the output window. (The right mouse button also brings up a limited menu for editing the output window.)

But, there is a way to get full editing capability. You can select, then cut or copy any material from the output window and paste it into an editing window (or into any other program, such as a word processor, that you might be using at the same time). The editing window then provides full editing capability, so you can place any annotation in the results that you like. You can save the contents of the editing window as an ordinary text file when you exit *LIMDEP*.

TIP: If you wish to extract from your output window a little at a time, one approach is to open a second editing window, and use it for the output you wish to collect. You may have several editing windows open at any time.

R2.10.3 Printing

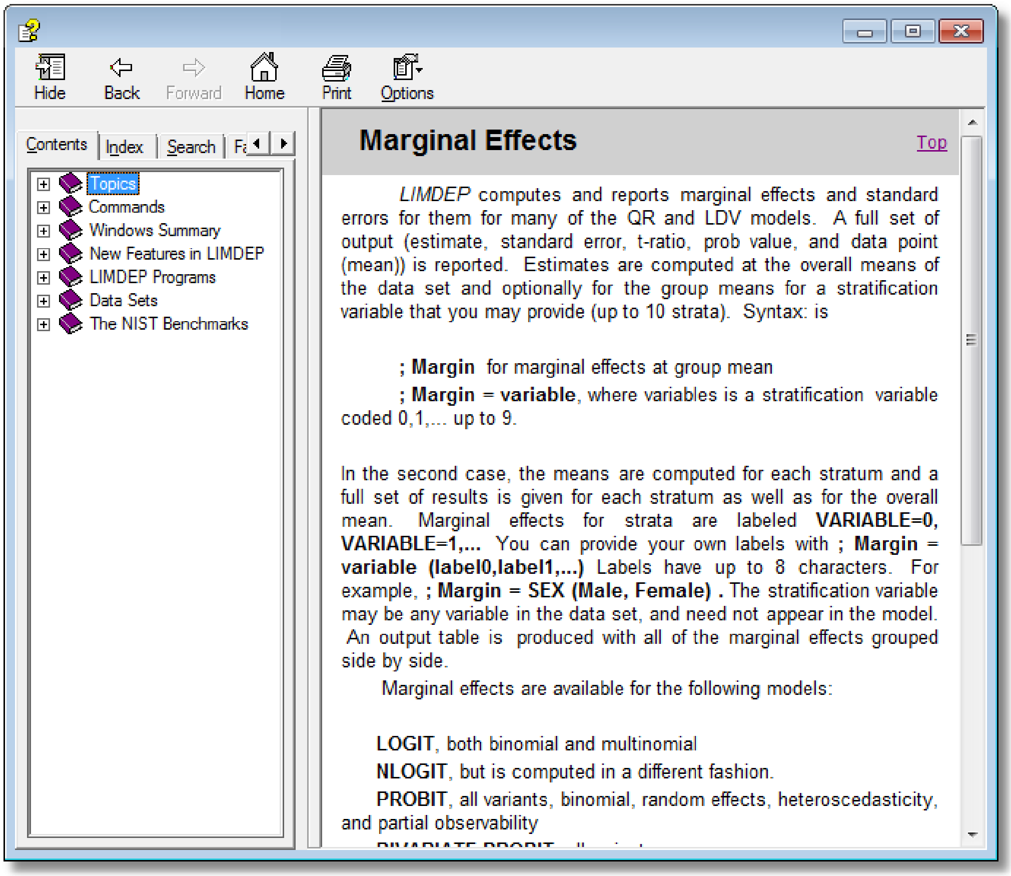
Printing with *LIMDEP* is handled by your Windows print manager. Also, you will generally do relatively little printing during your *LIMDEP* session, and, probably, relatively little printing with *LIMDEP* at all – most results will go to a file on your disk, or can be pasted into word processing programs that can be used to process results for final output.

The File menu does give you some control over how *LIMDEP* results are to be printed from the program. The option File:Page Setup generates a dialog box where you can adjust the page orientation and margins.

R2.11 Help

LIMDEP offers an extensive Help file. Select Help:Help Topics from the menu to bring up the help editor. *LIMDEP*'s Help file is divided into seven parts, or 'books,' as shown in Figure R2.24. In the first book, you will find a selection of *Topics* that discuss general aspects of operating the program. In Figure R2.24, for example, the Help material on Marginal Effects is displayed. The second book is the *Commands* list. This contains a list of the essential features and parts of all of *LIMDEP*'s commands. The third book contains an expanded version of the desktop summary that appears in Section R2.13. The fourth book contains descriptions of new features in *LIMDEP*. Finally, there are three books of useful ancillary material: a collection of *LIMDEP* programs, some of which appear in this manual, a collection of data sets that can be used for learning how to use *LIMDEP* and for illustrating the applications – these include the data sets used in the applications in this manual, and, finally, some of the National Institute of Standards accuracy benchmark data sets. The files in the last three books are also available in a resource folder created when *LIMDEP* is installed. The location for the folder is C:\LIMDEP11, and there are three subfolders, Data Files, Command Files, and Project Files.

NOTE: All sample data files referenced in the documentation, as well as many of the NIST datasets and sample command and project files may be found in these folders and also in the Help file books.



Marginal Effects

LIMDEP computes and reports marginal effects and standard errors for them for many of the QR and LDV models. A full set of output (estimate, standard error, t-ratio, prob value, and data point (mean)) is reported. Estimates are computed at the overall means of the data set and optionally for the group means for a stratification variable that you may provide (up to 10 strata). Syntax: is

; Margin for marginal effects at group mean

; Margin = variable, where variables is a stratification variable coded 0,1,... up to 9.

In the second case, the means are computed for each stratum and a full set of results is given for each stratum as well as for the overall mean. Marginal effects for strata are labeled **VARIABLE=0, VARIABLE=1,...** You can provide your own labels with **; Margin = variable (label0,label1,...)** Labels have up to 8 characters. For example, **; Margin = SEX (Male, Female) .** The stratification variable may be any variable in the data set, and need not appear in the model. An output table is produced with all of the marginal effects grouped side by side.

Marginal effects are available for the following models:

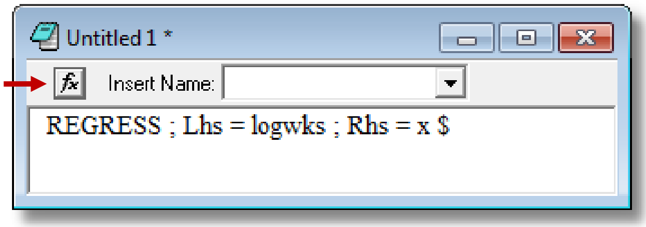
LOGIT, both binomial and multinomial

NLOGIT, but is computed in a different fashion.

PROBIT, all variants, binomial, random effects, heteroscedasticity, and partial observability

Figure R2.24 Help Books

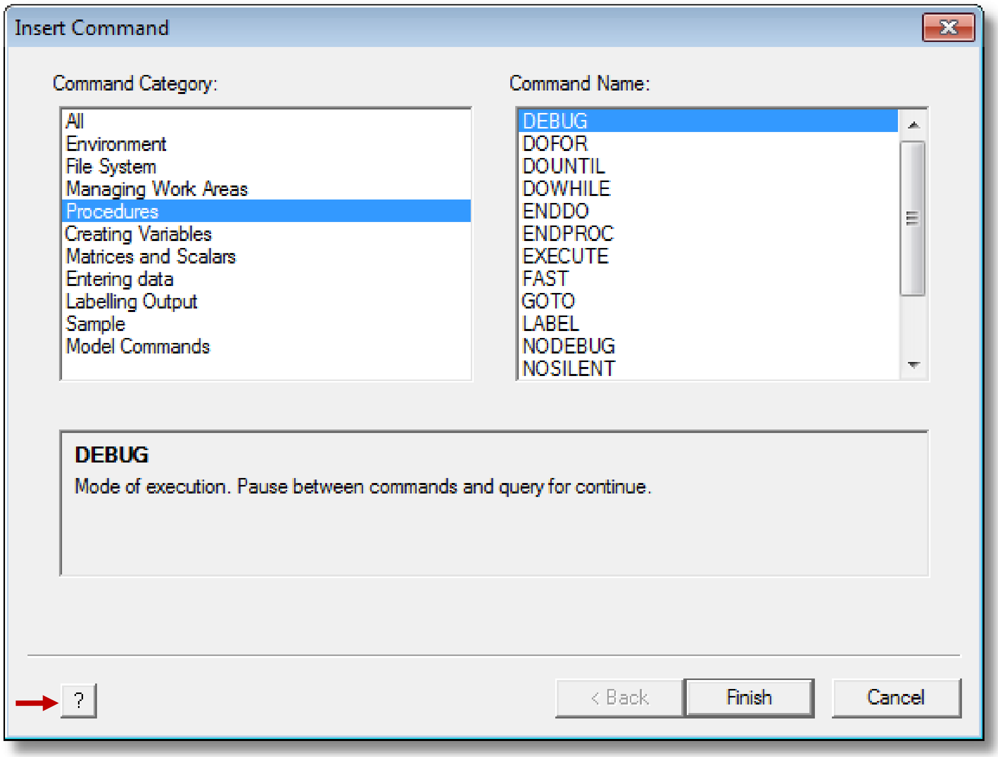
Many of *LIMDEP*'s features include context sensitive access to the Help file. You can access this information by clicking the ? button when available in a window. For example, the editing window includes a function button, f_x, to the left of the window, as shown in Figure R2.25.



REGRESS ; Lhs = logwks ; Rhs = x $

Figure R2.25 Editing Window with Function Button

Click the f_x button to open the Insert Command dialog box that allows you to insert any of the *LIMDEP* commands in the editing window. See Figure R2.26.



DEBUG

Mode of execution. Pause between commands and query for continue.

Figure R2.26 Insert Command Dialog Box

This dialog box offers a full list of the *LIMDEP* commands broadly grouped by function. Highlight a command category and a specific verb in that category. Then, click the ? button at the lower left corner of the window to open the Help file *Commands* section describing that specific command.

R2.12 Summary of Commands

This section will summarize the functions and commands available in *LIMDEP*. The listing given will suggest the range of procedures available. Most of the procedures listed here have numerous options, so this is merely an overview.

File System

CLOSE	Close an output file before opening a different output file.
LOAD	Retrieve saved data set to reactivate program. Use File:Open. (**LOAD ; File = filename $** may also be used.)
SAVE	Store all data currently active in a file; used with **LOAD**. Use File:Save.

Managing the Work Areas

DELETE	Delete variables. Clear space in the data work area. This can also be done by highlighting a variable in the project window and pressing the Del key. You can also delete matrices, scalars, and namelists this way.
LIST	Display variables on the screen. Inspect columns of data. This can also be done by double clicking a variable name in the project window to open the data editor. The data editor can also be opened by clicking the data editor button in the *LIMDEP* toolbar (grid/spreadsheet icon).
NAMELIST	Identify a list of variables with a name. The namelist editor can be opened from the project window by double clicking any namelist name.
RENAME	Change the names of one or more variables.
RESET	Delete all data of all types and restart session. This can also be done by selecting Project:Reset.

Creating and Executing Procedures

Procedures can be edited in a procedure editor by double clicking any procedure name in the project window. You can also begin entry of a new procedure in the procedure editor by selecting Run:New Procedure.

DOFOR	Execute a procedure for certain values of a variable.
DOUNTIL	Execute a procedure until a certain condition is true.
DOWHILE	Execute a procedure while a certain condition is true.
ENDDO	End of target procedure for **DOWHILE**, **DOFOR**, and **DOUNTIL**.
ENDPROC	End entry of commands in a procedure.
EXECUTE	Execute stored procedure. You can also execute a library procedure.
GO TO	Redirect the flow of execution of a set of commands.
LABEL	Mark a point in a set of commands. Use with **GO TO**.
PROC	Begin entry of commands in a procedure.
SILENT	Execute a procedure without displaying results.
NOSILENT	Turn off **SILENT** switch.
STRING	Define ‘macros’ for routines. Shorthand for a string of text.
LOCAL	Define certain matrices, variables and scalars that are local to the procedure rather than global in the general work areas.

Creating New Variables

CREATE	Transformations of variables. You can also enter transformations interactively. In the data editor, click the *right* mouse button and select New Variable from the menu. This opens a dialog box which allows you to enter **CREATE** commands interactively.
RECODE	Replace values of a variable with other values.
SORT	Sort a variable, possibly carrying others. (You can request a sort by highlighting the name in the project window, then right clicking.)

Manipulating Numeric Entities – Matrices and Scalars

CALCULATE	Compute scalar result.
MATRIX	Matrix algebra package.

These two procedures can be accessed from the main menu with Tools:Scalar Calculator or Matrix Calculator. Once in one processor, you can also switch directly to the other.

Entering and Documenting a Data Set

APPEND	Add additional observations to existing variables.
DATA	Create a text file that is embedded in the project. The text generally describes the data in the file. A previous usage of **DATA** was to access the data editor. The data editor may be opened by double clicking any variable name in the project window, selecting Project:Data Editor, or clicking the data editor button in the *LIMDEP* toolbar.
READ	Read a data set into the data work area from a file. You can also read a data set by entering the data editor, then clicking the *right* mouse button. In the menu, select Import Variables.
ROWS	Configure number of rows in data area. Use Project:Settings/Data Area.
WRITE	Write a data set in a disk file. You may also use Project:Export:Variables.

Labeling and Storing Statistical Output

REVIEW	Examine previous statistical results and create tables. Review can be reached by clicking any of the tables in the project window or by selecting Tools:Review Tables.
TABLE	Create tables of results in an output file.
TEXT	Send text to an output file.
TIMER	Display elapsed time for each model command. The switch can also be set with Project:Settings/Execution.
TITLE	Define page header for model commands.
TYPE	Send a message to screen and output file.
DISPLAY	Create a table of estimated model results from a vector of estimates and an estimated covariance matrix.
CLIST	Define a set of labels that can be used in several output functions.
LASTMODEL	Define a set of model results to be used in the **PARTIAL EFFECTS** program.

Defining the Sample to be Used for Estimation

SETPANEL	Global setting for a panel data set.
DATES	Define type of time series data, quarterly, yearly, etc. Use the command or Project:Settings/Data Type.
DRAW	Draw a random sample. Also allows bootstrap sampling.
INCLUDE	Add observations to sample. **INCLUDE** and **REJECT** can also be specified by using Project:Set Sample.
NOSKIP	Turn off **SKIP** switch. Can also be reached by Project:Settings/Execution. **SKIP** switch can also be turned on this way.
PERIOD	Define sample period for time series data. Can also be done with Project:Set Sample.
REJECT	Delete observations from sample.
SAMPLE	Specify observations in the sample by observation number.
SKIP	Set switch so *LIMDEP* automatically skips missing data.

Model Commands

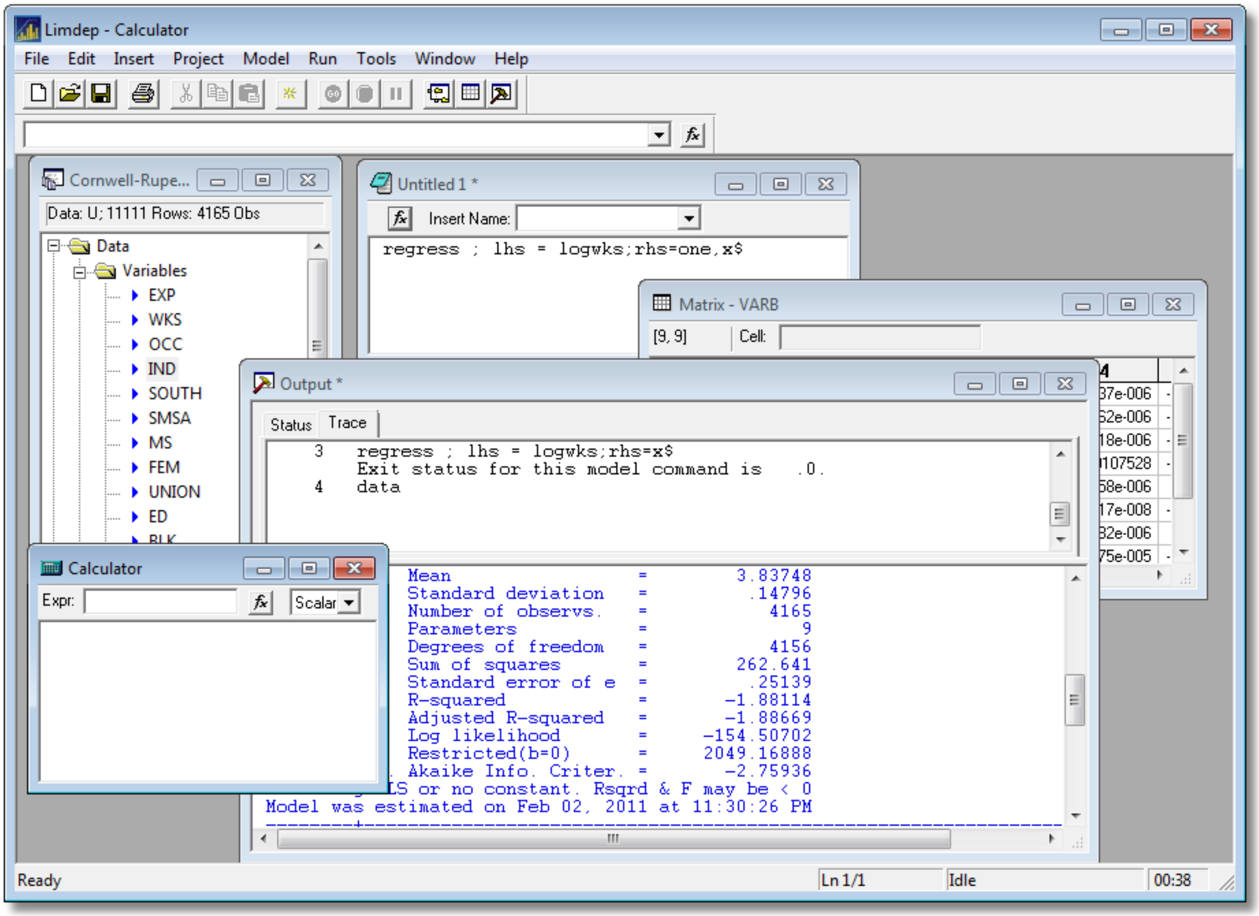
There are now well over 100 model commands supported in *LIMDEP* (and *NLOGIT*). A few of these are:

ARMAX	Box-Jenkins ARIMA models.
BIVARIATE PROBIT	Bivariate probit models.
CROSSTAB	Cross tabulation. Frequency counts and contingency tables.
DISCRETE CHOICE or **CLOGIT**	Random utility models.
DSTAT	Descriptive statistics.
FPLOT	Plot values of a function of a variable.
FRONTIER REGRESSION	Stochastic frontier.
GMME	Generalized method of moments estimation.
GOMPIT	Gompertz model for binary choice.
GROUPED DATA REGRESSION	Completely censored data.
HISTOGRAM	
MAXIMIZE	Maximize a user defined function.
MIMIC	Model for multiple indicators and multiple causes of a latent variable.
MINIMIZE	Minimize a function or compute nonlinear least squares estimates.
MPLOT	Plot elements of one matrix against those of another.
MPROBIT	Multivariate probit model.
NLSQ	Nonlinear least squares regression.
ORDERED PROBIT	Ordered probit or logit models.
PROBIT	Univariate probit model.
TOBIT	Censored regression.
QREG	Quantile regression.
REGRESS	Linear least squares regression.
SELECTIVITY	Sample selection models.
SPECTRAL	Plot and compute spectral density function.
SURE	Seemingly unrelated and multivariate regression.
3SLS	Three stage least squares.

R2.13 Summary of the *LIMDEP* Desktop

When you operate *LIMDEP*, you will generally be using the set of windows shown in Figure R2.27, which is a somewhat abbreviated composite of the various features that you will find on your screen. This section of the documentation will briefly describe the different parts of the desktop. In the sections to follow, we will describe

- The three main windows, plus the calculator window,
- The main menus shown at the top of the screen,
- The toolbar shown below the main menus,
- The command window or command bar below the toolbar,
- The correspondence between the menu items and the commands listed earlier.



```
regress ; lhs = logwks;rhs=one,x$
```

```
   3    regress ; lhs = logwks;rhs=x$
        Exit status for this model command is    .0.
   4    data
```

```
Mean                     =         3.83748
Standard deviation       =          .14796
Number of observs.       =            4165
Parameters               =               9
Degrees of freedom       =            4156
Sum of squares           =         262.641
Standard error of e      =          .25139
R-squared                =        -1.88114
Adjusted R-squared       =        -1.88669
Log likelihood           =      -154.50702
Restricted(b=0)          =      2049.16888
Akaike Info. Criter.     =        -2.75936
LS or no constant. Rsqrd & F may be < 0
Model was estimated on Feb 02, 2011 at 11:30:26 PM
```

Figure R2.27 *LIMDEP* Desktop

R2.13.1 The *LIMDEP* Windows

There are three main windows on the desktop. Operation begins in the project window, which is at the upper left of the desktop. The project window contains a complete inventory of the data, matrices, procedures, and so on, that you have created during your session. By clicking the different topics in the project window, you can review the variables, matrices, etc. that exist at any time. In addition, you can launch many different operations from the project window.

Most of your command input is done from the editing window, which is at the upper center of the desktop. Model commands and data manipulation commands will usually be placed on the screen in this window, then executed by one of several methods. The simplest way to proceed until you are ready to use the more advanced features is to place the commands you wish to execute in the window, highlight them, then click the green GO button. There are many options available for editing in this window, including two of the main menus, Edit and Insert.

Your statistical results and a trace of your session are accumulated in your output window. This is the larger window at the lower right of the desktop. We have compressed it for the display. The output window is a split screen. The statistical results are shown in the lower half. The upper part has two tabs. If you choose Trace, a continuing trace of your commands, diagnostics and error messages that your commands produce, and other useful information are listed in the top half of the window. This information is also saved at the end of your session in the *LIMDEP* trace file, trace.lim. If, instead, you have chosen the Status tab, the top half of the window will display certain technical output generated during model estimation, such as values of the log likelihood, convergence criteria, and timing information. The trace will continue to accumulate in the background.

The smaller window at the left of the desktop is a calculator window. You can open a calculator or matrix window by selecting Tools:Scalar Calculator or Tools:Matrix Calculator. You may, in fact, have more than one of these open at any time, though typically, using just one is best. The calculator window is an interactive session with **CALCULATE** and/or **MATRIX**. Results that you wish to obtain interactively rather than as part of a command in a procedure, can be obtained by entering one of these windows. The window contains its own input field, at the top, labeled 'Expr' for expression. You can enter any valid **CALC** or **MATRIX** command in this field. The result will be shown in the lower field. The window shown in the figure is being used for **CALC**. You can switch over to **MATRIX** by clicking the ▼ button at the right of the top row, next to the window containing 'Scalar.' Thus, you can accumulate both matrix and scalar results in this window. (Matrix results are shown as an object, rather than the full matrix, itself. By clicking the object, you can enter a display of the matrix, itself.) Finally, the f_x button will open the Insert Function dialog box containing a selection of functions that you can insert into your expressions for both **CALC** and **MATRIX**.

R2.13.2 The Main Menus

The nine menus at the top of the desktop provide the following functions:

File Menu

File		Opens and closes files for your *LIMDEP* session.
New...	Ctrl+N	Opens a new editing window (Text/Command Document) or a new project window (Project).
Open	Ctrl+O	Opens a file into project, editing, or output window.
Close		Closes the active window. (You are asked for confirmation.)
Save	Ctrl+S	Saves the active window (project, editing, output).
Save As...		Same as Save.
Save All		Saves all windows.
Open Project...		Opens window to find a project file. Same as **LOAD ; File = filename $**
Save Project As...		Same as **SAVE ; File = filename $**
Close Project		Same as **RESET $**
Page Setup...		Sets up printing for output window – size, portrait/landscape, margins, printer identity, port. Also sets up graphics output.
Print Preview		Displays on the screen what printed output will look like.
Print...	Ctrl+P	Standard Windows print operation, to print output window.
1 Names of previous 2 .lim files used (data, 3 command, or output 4 files)		Opens this .lim file. Names of previous command, data or output files that were opened by file name will appear here, in the order in which they were used.
5 Names of recent 6 project files used 7 8		Opens the most recently used project file. Lists up to four recent projects.
Exit	Alt+F4	Exits *LIMDEP*. You are queried to save project, output, and editing windows (.lpj and .lim file types) and any calculator, matrix, and plotting windows that are still open.

Edit Menu

Edit		Standard editing features when in the editing window.
Undo	Ctrl+Z	Undoes the last operation, for example, replaces deleted text.
Cut	Ctrl+X	Removes selected text and puts it in the clipboard.
Copy	Ctrl+C	Copies selected text to the clipboard, without deletion.
Paste	Ctrl+V	Copies last cut or copied text to insertion point.
Clear Del		Erases selected text in the output window or the trace window.
Delete		(Not used.)
Select All	Ctrl+A	Selects (highlights) entire editing or output window.

These editing features also operate in the output window. Del (key) also deletes a selected item in the project window.

Include Observations	Ctrl+add	Sets up an **INCLUDE** command in a window.
Reject Observations	Ctrl+subtract	Sets up a **REJECT** command in a window.

Find	Ctrl+F	Finds a particular character string in the editing window.
Find Next	F3	Repeats the last find operation. Finds next occurrence.
Replace		Replaces a character string with another character string.
Go To...	Ctrl+G	Moves insertion point to a particular line (by line number).
Object		(Not used.)

Insert Menu

Insert	Inserts items at the insertion point in the editing window. These features operate on the text editing window to help build commands. You can insert verbs (command names), file paths, or entire ASCII text files at the insertion point.
Command	Inserts a command (verb only) at the insertion point. A window presents the list of verbs to select from.
File Path...	Inserts the full path to a file on your hard disk at the insertion point. Used in commands such as **OPEN** and **WRITE**.
Text File...	Inserts a text file at the insertion point. You can put the entire contents of a file in the editing window. The file can be added to the existing text or put into an empty screen for editing or for execution as an input file.

Rows	Opens up rows at insertion point in the data editor (not the text editor). This puts one or more empty rows at the insertion point in the data editor. Data are pushed off the stack.
Columns	(Not used.)

Item into Project…	Inserts an item into project.
Variable	**CREATE** a new variable.
Namelist	**NAMELIST** command for a new namelist.
Matrix	**MATRIX** command for a new matrix.
Scalar	**CALCULATE** command for a new scalar.
Procedure	Enters a procedure editor to create a procedure.

Project Menu

Project	Use for several functions of managing the data as well as a number of other features of the program.
Settings…	Offers a window with tabs for parameter and switch settings.
Data Area	Memory allocation: chooses number of cells in data area. Dimensions: chooses number of rows in data area. Same as **ROWS** command.
Data Type	Observations: Chooses data type, cross section (undated) or time series (dated). Same as **DATES** command. Initial Date: yearly, quarterly, or monthly if time series data.
Execution	Models: Turns **SKIP** switch on or off.
Output	Turns **SILENT** switch on or off to suppress output. Turns **TIMER** switch on or off to display time used in estimation.
New ▸	Creates five new entities (variables, etc.). These open windows that operate like the individual commands, but in interactive, rather than command mode.
Variable…	Interactive create window.
Namelist…	Interactive namelist editor.
Matrix…	Interactive matrix window.
Scalar…	Interactive calculator window.
Procedure…	Interactive procedure editing window.
Import ▸ Variables	Finds a data file by opening a search window. Same as **READ.**
Export ▸ Variables	Writes variables in a data file. Formats include .xls, .wk1, .dat, and binary (.bin). Same as **WRITE**.
Data Editor	Goes to data editor – spreadsheet style display.
Sort Variable...	Sorts a variable, carrying one or more other variables. Same as **SORT ; Lhs = variable to sort ; Rhs = variables to carry $; Ascending** optional.
Set Sample ▸	Sets the current sample.
All	Same as **SAMPLE ; All $**
Range...	Observation rows: Queries for observations to be in the current sample. Same as **SAMPLE ; Range $** for cross section data or **PERIOD ; First - Last $** for time series data.
Include...	Queries for **; New** and enters command (expression). Same as **INCLUDE ; Expression $** with **; New** optional.
Reject...	Queries for **; New** and enters command (expression). Same as **REJECT ; Expression $** with **; New** optional.
Draw...	Draws random samples from your data set. Same as **DRAW ; Nobs [; Replacement] $**
Reset...	Erases all variables and matrices. Same as **RESET $**

Model Menu

Model is used to invoke a dialog which helps beginning users set up model definitions. The initial menu offers a grouping of the available *LIMDEP* modeling frameworks. Each of these contains a number of more specific model commands. The complete Model menu and specific Linear Models options are shown in Figure R2.28.

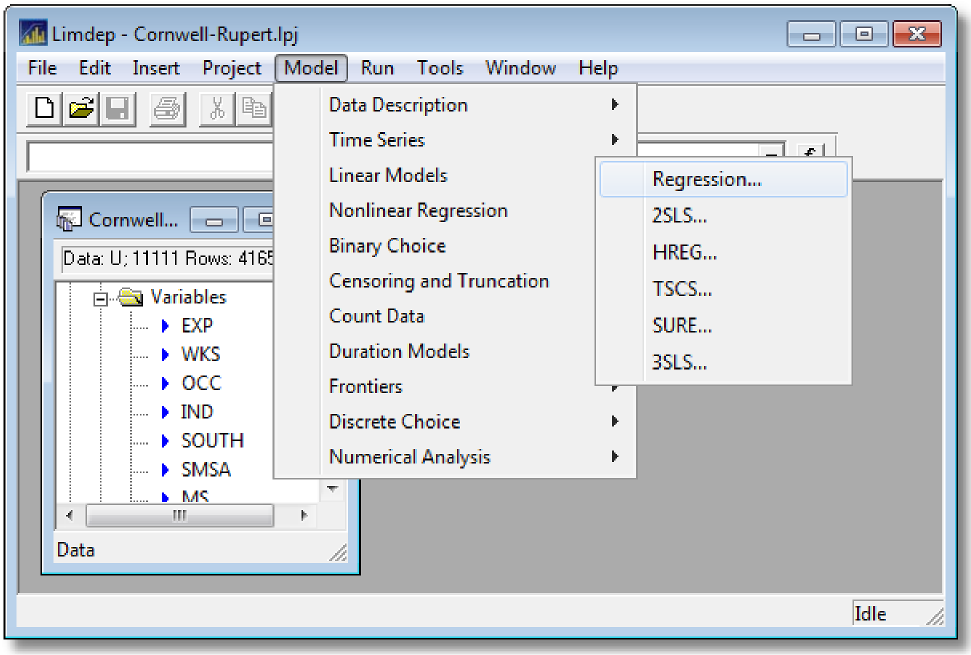


Figure R2.28 Model Menu and Linear Model Options

Each specific model has associated with it a ‘command builder’ which you can use to specify a model in a dialog box. The dialog then produces two or three windows which offer the options available for that modeling framework. For example, the Regression command builder shown in Figure R2.29 offers main, options, and output selections. The main and options dialog boxes (pages) are shown below. The command builder will be discussed further in Chapter R8.

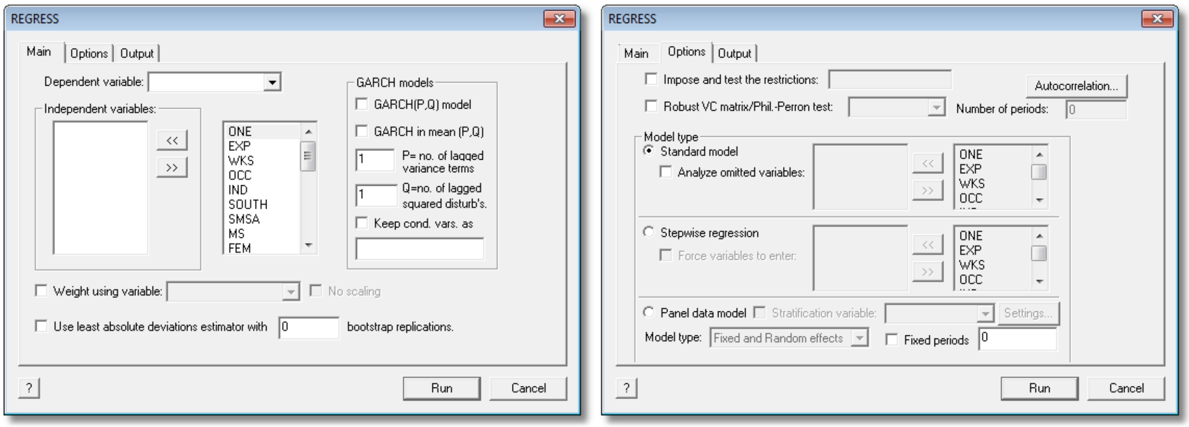


Figure R2.29 Regression Command Builder Dialog Boxes

Run Menu

Run	Executes groups of commands highlighted in the editing window.
Run Ctrl+R	Executes highlighted commands. Same as GO button.
Run Multiple Times...	Executes highlighted lines more than one time.
Run File…	Executes the commands in a file. Same as **OPEN ; Input = filename $** Queries for filename.
Run Procedure	Executes a procedure stored in the procedure library.
Stop Running Ctrl+Break	Interrupts input of a file or execution of commands from the editing window.
New Procedure	Opens procedure editor. Saves procedure on exit. Procedures may be named and have adjustable, replaceable parameters.

Tools Menu

Tools	Interactive execution of certain commands such as **CALC** and **MATRIX.**
Scalar Calculator	Opens a calculator window for scalar calculations. Same as **CALCULATE** command. You can toggle to the matrix calculator from this window. Results are shown on the screen.
Matrix Calculator	Opens a calculator window for matrix calculations. Same as **MATRIX**. You can toggle to the scalar calculator window from this window. Results are displayed on the screen.
Review Tables...	Constructs tables of statistical results. Model commands that contain **; Table = name** accumulate results in the model stack that are reviewed here and can be grouped in tables in the output window.
Options...	Several settings for program execution and display.
View	Displays toolbar at top of desktop.
	Displays program status bar at bottom of desktop.
	Displays command bar at top of desktop.
Editor	Chooses font for editing and output windows.
	Automatic word selection handles highlighting in editing window.
	Show Editor Tool Bar displays $\boldsymbol{f_x}$ button and small Insert Name window at top of editing window.
Projects	Main memory allocation for data area. Same as **RESET**.
Execution	Output window moves to front during execution.
	Shows error message dialog boxes during execution.
	Preferences: faster execution or greater user interface responsiveness.
Trace	Indicates where to save trace file trace.lim.

Window Menu

Window	Handles display of windows on the desktop.
Split	Sets sizes of parts of split output window.
Cascade	Arranges active windows in cascade display format.
Tile	Arranges active windows in tile display format.
Arrange Icons	Arranges icons at bottom of desktop.
Output	Activates output window.
1 Names of recent open 2 windows (project, editing, 3 output, calculator, etc.) 4	Activates any of these windows.

Help Menu

Help	
Help Topics	Activates *LIMDEP* Help program.
Tip of the Day…	Suggestions for operating *LIMDEP*.
LIMDEP Web Site…	If you have a web browser active, sends web browser to *LIMDEP* home page.
About LIMDEP…	Information box.

Other menus include the following:

Icon Menu

Clicking the icon at upper left corner of an active window produces the following menu:

Restore		Restores window after it has been minimized.
Move		Repositions window on screen.
Size		Changes size of window.
Minimize		Reduces window to icon on taskbar.
Maximize		Expands window to fill desktop.
Close	Ctrl+F4	Closes window.
Next		Moves cursor to next open window.

Right Mouse Button Menu

Clicking the right mouse button when mouse cursor is not in a window opens this menu:

New…	Same as File:New.
Open…	Same as File:Open.
Project	Activates project window.
Data Editor	Activates data editor window.
Output	Activates output window.
Options…	Same as Tools:Options.

R2.13.3 The *LIMDEP* Toolbar

There are fourteen tools shown in the *LIMDEP* toolbar (located below the main menu). If the toolbar is not showing on your screen, select the Tools:Options/View tab, then turn on the Display Tool Bar option.

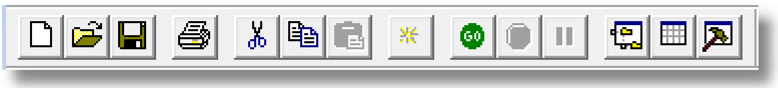


Figure R2.30 The *LIMDEP* Toolbar

These are equivalent to certain other menu entries, as follows. The buttons are listed below in the order from left to right:

1. Open a new editing or project window. (File:New)
2. Open an existing file. (File:Open)
3. Save. (File:Save)
4. Print. (File:Print)
5. Cut selection to clipboard. (Edit:Cut)
6. Copy selection to clipboard. (Edit:Copy)
7. Paste selection from clipboard to insertion point. (Edit:Paste)
8. Insert an item into the project window. (Insert:Item into Project…)
9. Execute selected commands. (Run:Run)
10. Stop commands. (Run:Stop Running)
11. Pause commands (from file or editing window).
12. Activate project window.
13. Activate data editor.
14. Activate (and open if necessary) output window.

R2.13.4 The Command Bar

The command bar (or command window) appears below the toolbar. If the command bar is not showing on your screen, select the Tools:Options/View tab, then turn on the Display Command Bar option. By placing the insertion point in this window, you can enter your commands one at a time. This is for one line commands. The command is submitted when you press Enter. The window accumulates a menu of your commands, which is recalled by clicking the ▼ button at the right end of the command window.

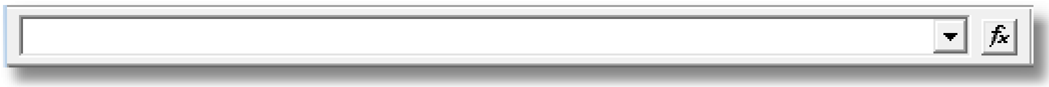


Figure R2.31 The *LIMDEP* Command Bar

R2.13.5 Commands and Menu Items

Several commands are equivalent to the menu items listed earlier. Also, in some cases, there is more than one menu item which corresponds to the particular command. The correspondences are listed below.

CALC	Tools:Scalar Calculator or toggle from the matrix calculator window
CREATE	In data editor, right button, New variable. Project:New Variable
DATES	Project:Settings/Data Type
DELETE	Highlight name in project window then press Del
IMPORT	Project:Import/Variables
INCLUDE	Project:Set Sample/Include or Edit:Include Observations
LOAD	File:Open or File:Open Project (any project shown in submenu)
MATRIX	Tools:Matrix Calculator or toggle from the scalar calculator window
NAMELIST	Double click existing namelists in project window to edit a namelist, or Project:New/Namelist
OPEN	Output file, automatically saves output window at exit
PERIOD	Project:Set Sample/Range (set dates for time series first)
READ	Project:Import/Variables
REJECT	Project:Set Sample/Reject, Edit:Reject Observations
RESET	Project:Reset
REVIEW	Tools:Review Tables
SAMPLE	Project:Set Sample/Range
SAVE	File:Save, Save As, Save Project As
SILENT	Project:Settings/Output, disable Show output
SKIP/NOSKIP	Project:Settings/Execution, enable/disable Skip missing data
STRING	Double click any string in project window to open string editing window
TIMER	Project:Settings/Output, enable Show model execution time in output
WRITE	Project:Export/Variables

R3: Importing and Reading Data Files

R3.1 Importing and Reading Data

Step one of your analysis is getting your data into *LIMDEP*. *Externally created data*, such as public data bases and data sets assembled from external sources are imported via disk files, downloaded from the internet or, for very small data sets, prepared for *LIMDEP* using the text and data editors. *Internally created data* for simulations and experiments are produced using *LIMDEP*'s random number generators. This chapter is about importing externally created data. The **CREATE** command described in Chapter R4 is used to produce internally created data.

There are two very similar operations used for getting your data into the program, the **IMPORT** and **READ** commands. **IMPORT** is used for standard forms of ASCII data files, including CSV files such as those created with Microsoft *Excel*. Data files sometimes come in other forms, such as binary files, files with other structures, or files produced by other programs. *LIMDEP* can read many kinds of files that do not fit the standard format. The operation for these files is **READ**. **IMPORT** will be described first in this chapter in Section R3.2. **READ** will follow in Section R3.5.

In almost all cases, you will import your data into a data area that is created for storage of the data while they are being analyzed. There are two alternative forms of importing that you may use on occasion. The **APPEND** operation is used when you wish to read additional observations on variables that you have already imported. **APPEND** is described in Section R3.10. The **MERGE** operation is used to interleave two files for a panel data set, in which one contains observations on variables that vary within a single 'group,' and a second which contains variables that are only observed once for each individual. **MERGE** is described in Chapter R5.

Data are stored in an area of memory that we will refer to as the 'data array.' The number of cells in this area may vary as you use *LIMDEP* – you can change its size if you need to. The initial setting of the data array is 5,000,000 cells, which is large enough that you will probably never need to adjust it. Procedures for doing so for very large data sets are noted in Section R3.4. Once the data have been imported, no distinction is maintained between integer and noninteger values, and there is no need to maintain any consideration of how many digits a number contains. All numbers are treated the same. Operations that are based on integer values are handled internally.

LIMDEP provides two commands, **EXPORT** and **WRITE** for creating data files to be exported to other programs. **EXPORT** is used to create a standard ASCII, CSV file. **WRITE** is used for some other formats. These commands are discussed in Section R3.9.

R3.2 Import a Standard Formatted ASCII File

IMPORT can be used to input a standard ASCII data file with the following form:

- The file is ASCII text. You can see the contents in a text editor or word processor.
- Observations on sets of variables are on a single line in the file with items separated by commas.
- Variable names, if given in the data file, are provided in the first row, separated by commas, and must appear on a single line only
- Observation labels, if provided, are in the first column and include a column heading.
- If observation labels are provided, the numeric data must be given on one line.

NOTE: CSV files such as those created with Microsoft *Excel* follow this format and can be input with **IMPORT**. (When a CSV file is viewed in *Excel*, the data and variables appear in cells, rather than separated by commas.)

The basic case of a standard data file is a rectangular ASCII text shown in Figure R3.1. This example has variable names in a single row at the top of the file.

```
ID  Year  Age,    Educ
1   1960   23      16
2,  1975,  44,     12.5
3   1990   14      11.5
4   1993 missing  20
```

Figure R3.1 Sample Data File

To import a data file of this form, you need only tell *LIMDEP* where it is. You can use menu options or the command mode. For the menu option, select Project:Import, then select Variables, to open the Import dialog box, as shown in Figures R3.2 and R3.3.

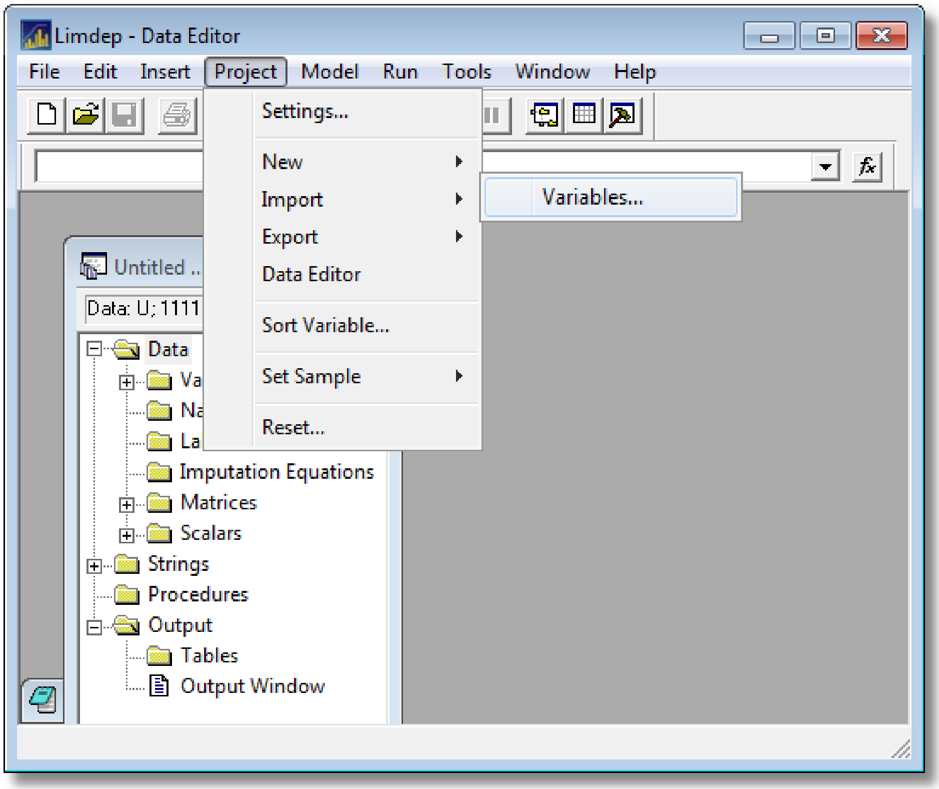
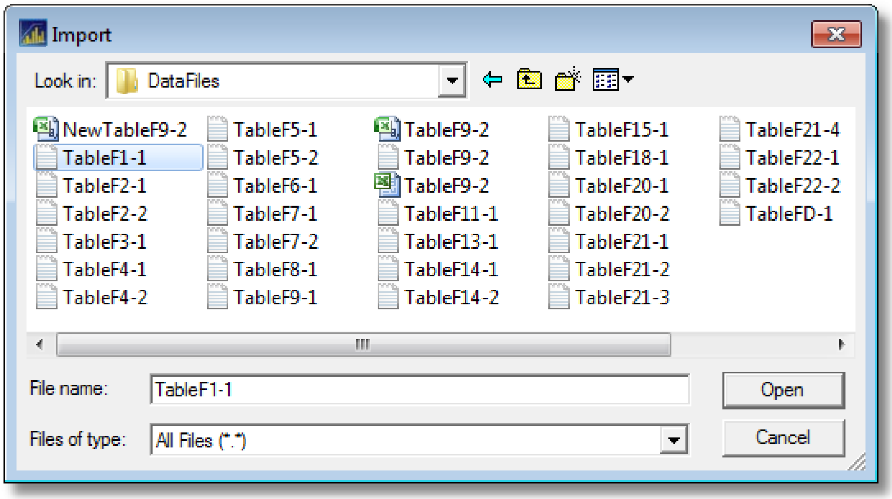


Figure R3.2 Project:Import/Variables Menu

In the Import dialog box, select All Files (*.*) in the 'Files of type' window, then locate and select your data file and click Open.



Look in: DataFiles

File name: TableF1-1

Files of type: All Files (*.*)

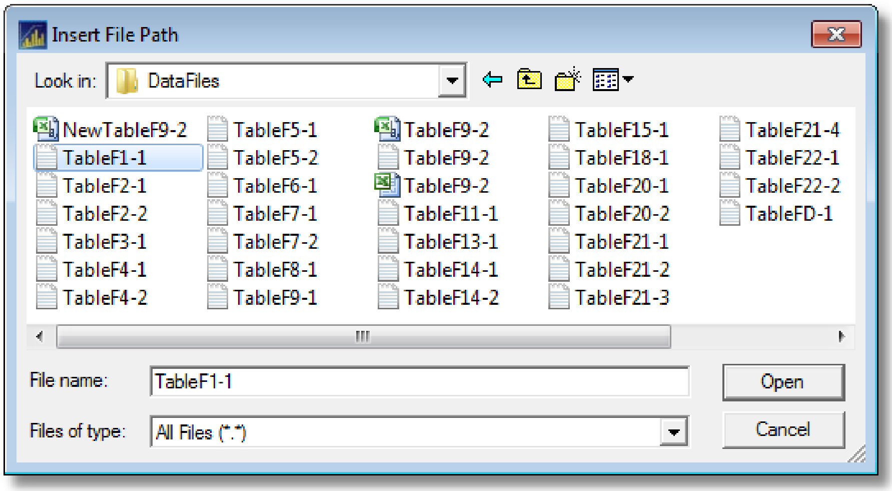
Figure R3.3 Import Dialog Box

The **IMPORT** command for importing a data file is

IMPORT ; File = … < the name of the file, including the path>… $

The **IMPORT** command is submitted from the text editing window. As described in Chapter R2, to open an editing window, click File:New, then Text/Command Document, then OK. Once you have entered your commands, just highlight the commands and click GO on the toolbar to submit your commands to *LIMDEP*.

TIP: When you use the **IMPORT** command, you need to specify the full path to a file. Sometimes this is hard to locate. You can obtain the full path to a file by using Insert:File Path. The file path will be inserted where the cursor is in the open editing window. Select Insert:File Path to open a dialog box where you can locate the file you wish to open. When you click Open, the full path to the file will be placed in double quotes in the editor window. An example is shown Figures R3.4 and R3.5.

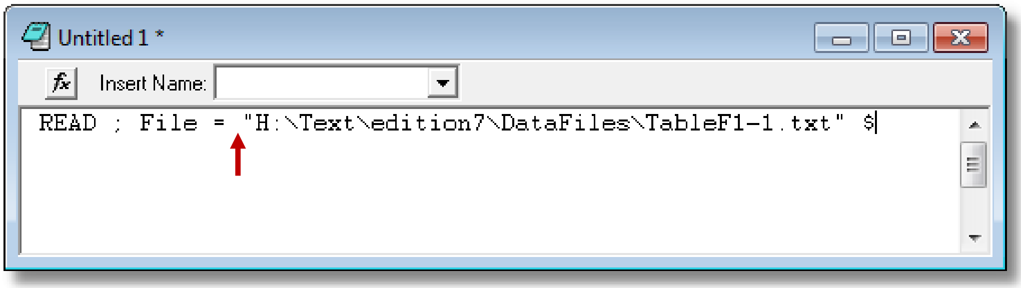


Look in: DataFiles

File name: TableF1-1

Files of type: All Files (*.*)

Figure R3.4 Insert File Path Dialog Box



```
READ ; File = "H:\Text\edition7\DataFiles\TableF1-1.txt" $
```

Figure R3.5 Editing Window with Insert File Path

There is no other information that needs to be provided with the **IMPORT** command. When you read a file of this type, *LIMDEP* determines the number of variables to be read by counting the number of names that appear in the first line of the file. The number of observations in the file is determined by reading until the end of the file is reached. The sample data set shown in Figure R3.1 illustrates several degrees of flexibility.

- Variable names in a file may be separated by spaces and/or commas or tabs.
- Names need not be capitalized in the file. *LIMDEP* will capitalize them as they are read.
- The numbers need not be lined up in neat columns in a data file.
- Values in the data set may be separated by spaces and/or commas or tabs.
- There is no need to distinguish numeric types, integers (44) and reals (12.5).
- Missing values in a data set may be indicated by anything that is not a number. (But, they must be indicated by *something*. A blank is not understood to be a missing value.)
- If the file contains observation labels with the data, the number of variables is automatically adjusted. See Section R3.2.1 for details.

NOTE: You can import a data file that does not contain variable names. See Section R3.2.3.

R3.2.1 Observation Labels and Variable Names in the Data File

Data files often contain observation labels as well as variable names. The data set in Figure R3.6 shows the typical arrangement.

```
State           ValueAdd     Capital     Labor        NFirm
Alabama          126.148       3.804    31.551           68
California      3201.486     185.446   452.844         1372
Connecticut      690.670      39.712   124.074          154
Florida           56.296       6.547    19.181          292
Georgia          304.531      11.530    45.534           71
... (20 more observations)
```

Figure R3.6 Data File with Observation Labels

Use the **IMPORT** command or menu option exactly as before. The observation labels will be noticed and read separately. The following are required:

- The labels must appear in the first column. If they appear in a different column, then you will use the **READ** command to import the data. **READ** is discussed in Section R3.5.
- The labels column is an extra column in the data. It is not a variable.
- If there are names in the first line of the data file, then you must include a name for the labels. Note that the name 'State' is used for the labels, not for one of the variables.
- The labels must not contain spaces. 'West Virginia' is not a valid label – it will ultimately translate to 'West' for the label and a missing value for the first variable.
- Data may not be 'transposed.' (See Section R3.5.5.)
- The file may not be 'formatted.' (See Section R3.5.7.)
- The maximum number of observations in a labeled file is 65,536.
- Labels may contain up to 16 characters.

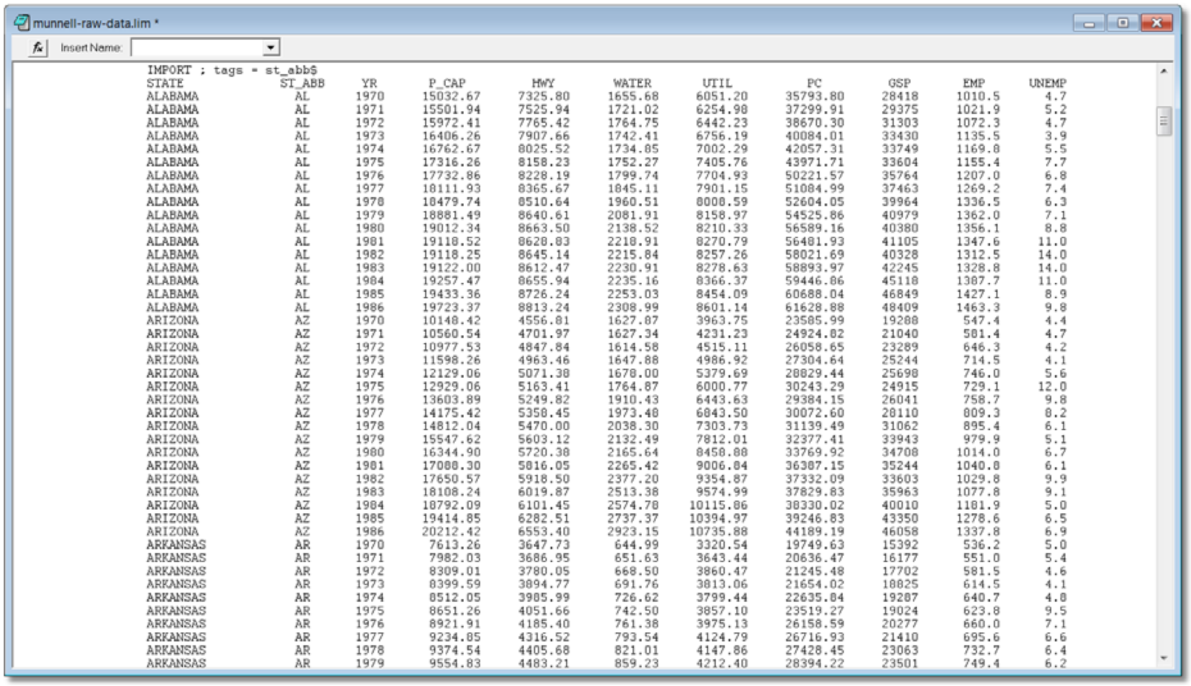
R3.2.2 Alphabetic Tags for Observations

The observation labels described above are used in some places to label the data in a list. See, for example, Figure R3.13 where the labels identify the rows of data in the display of the data editor. Labels are not actually used in any operations that you do with your data. Your data may also contain a set of alphabetic 'tags' that you can use for example to specify which observations to use in estimating a model or how to compute a transformed variable. Tags are imported or read with the data. Here is an example: This data set contains 816 observations, 17 yearly observations on each of the 48 continental United States. Figure R3.7 shows how to use **IMPORT** to read the data. (We are using the text editor as the input file. See Section R3.6 for discussion of this method.) The command is

IMPORT ; tags = ST_ABB $

Notice that the first 'variable' in the data is the state name. This is the observation label that will be used, for example, in the data editor. The **IMPORT** command identifies the column named ST_ABB (state abbreviation) as a set of tags. The tags will be read as a special kind of variable (alphabetic) in the data set. Observation tags will be useable in several operations. For example, after reading these data, you might want to compute a regression using only the data for ALABAMA which you could do with

REGRESS **; If [[ST_ABB] = AL] ; … (specification of the model) $**



```
IMPORT ; tags = st_abb$
STATE              ST_ABB    YR      P_CAP       HWY      WATER      UTIL        PC      GSP      EMP    UNEMP
ALABAMA              AL     1970   15032.67   7325.80   1655.68   6051.20   35793.80   28418   1010.5     4.7
ALABAMA              AL     1971   15501.94   7525.94   1721.02   6254.98   37299.91   29375   1021.9     5.2
ALABAMA              AL     1972   15972.41   7765.42   1764.75   6442.23   38670.30   31303   1072.3     4.7
ALABAMA              AL     1973   16406.26   7907.66   1742.41   6756.19   40084.01   33430   1135.5     3.9
ALABAMA              AL     1974   16762.67   8025.52   1734.85   7002.29   42057.31   33749   1169.8     5.5
ALABAMA              AL     1975   17316.26   8158.23   1752.27   7405.76   43971.71   33604   1155.4     7.7
ALABAMA              AL     1976   17732.86   8228.19   1799.74   7704.93   50221.57   35764   1207.0     6.8
ALABAMA              AL     1977   18111.93   8365.67   1845.11   7901.15   51084.99   37463   1269.2     7.4
ALABAMA              AL     1978   18479.74   8510.64   1960.51   8008.59   52604.05   39964   1336.5     6.3
ALABAMA              AL     1979   18881.49   8640.61   2081.91   8158.97   54525.86   40979   1362.0     7.1
ALABAMA              AL     1980   19012.34   8663.50   2138.52   8210.33   56589.16   40380   1356.1     8.8
ALABAMA              AL     1981   19118.52   8628.83   2218.91   8270.79   56481.93   41105   1347.6    11.0
ALABAMA              AL     1982   19118.25   8645.14   2215.84   8257.26   58021.69   40328   1312.5    14.0
ALABAMA              AL     1983   19122.00   8612.47   2230.91   8278.63   58893.97   42245   1328.8    14.0
ALABAMA              AL     1984   19257.47   8655.94   2235.16   8366.37   59446.86   45118   1387.7    11.0
ALABAMA              AL     1985   19433.36   8726.24   2253.03   8454.09   60688.04   46849   1427.1     8.9
ALABAMA              AL     1986   19723.37   8813.24   2308.99   8601.14   61628.88   48409   1463.3     9.8
ARIZONA              AZ     1970   10148.42   4556.81   1627.87   3963.75   23585.99   19288    547.4     4.4
ARIZONA              AZ     1971   10560.54   4701.97   1627.34   4231.23   24924.82   21040    581.4     4.7
ARIZONA              AZ     1972   10977.53   4847.84   1614.58   4515.11   26058.65   23289    646.3     4.2
ARIZONA              AZ     1973   11598.26   4963.46   1647.88   4986.92   27304.64   25244    714.5     4.1
ARIZONA              AZ     1974   12129.06   5071.38   1678.00   5379.69   28829.44   25698    746.0     5.6
ARIZONA              AZ     1975   12929.06   5163.41   1764.87   6000.77   30243.29   24915    729.1    12.0
ARIZONA              AZ     1976   13603.89   5249.82   1910.43   6443.63   29384.15   26041    758.7     9.8
ARIZONA              AZ     1977   14175.42   5358.45   1973.48   6843.50   30072.60   28110    809.3     8.2
ARIZONA              AZ     1978   14812.04   5470.00   2038.30   7303.73   31139.49   31062    895.4     6.1
ARIZONA              AZ     1979   15547.62   5603.12   2132.49   7812.01   32377.41   33943    979.9     5.1
ARIZONA              AZ     1980   16344.90   5720.38   2165.64   8458.88   33769.92   34708   1014.0     6.7
ARIZONA              AZ     1981   17088.30   5816.05   2265.42   9006.84   36387.15   35244   1040.8     6.1
ARIZONA              AZ     1982   17650.57   5918.50   2377.20   9354.87   37332.09   33603   1029.8     9.9
ARIZONA              AZ     1983   18108.24   6019.87   2513.38   9574.99   37829.83   35963   1077.8     9.1
ARIZONA              AZ     1984   18792.09   6101.45   2574.78  10115.86   38330.02   40010   1181.9     5.0
ARIZONA              AZ     1985   19414.85   6282.51   2737.37  10394.97   39246.83   43350   1278.6     6.5
ARIZONA              AZ     1986   20212.42   6553.40   2923.15  10735.88   44189.19   46058   1337.8     6.9
ARKANSAS             AR     1970    7613.26   3647.73    644.99   3320.54   19749.63   15392    536.2     5.0
ARKANSAS             AR     1971    7982.03   3686.95    651.63   3643.44   20636.47   16177    551.0     5.4
ARKANSAS             AR     1972    8309.01   3780.05    668.50   3860.47   21245.48   17702    581.5     4.6
ARKANSAS             AR     1973    8399.59   3894.77    691.76   3813.06   21654.02   18825    614.5     4.1
ARKANSAS             AR     1974    8512.05   3985.99    726.62   3799.44   22635.84   19287    640.7     4.8
ARKANSAS             AR     1975    8651.26   4051.66    742.50   3857.10   23519.27   19024    623.8     9.5
ARKANSAS             AR     1976    8921.91   4185.40    761.38   3975.13   26158.59   20277    660.0     7.1
ARKANSAS             AR     1977    9234.85   4316.52    793.54   4124.79   26716.93   21410    695.6     6.6
ARKANSAS             AR     1978    9374.54   4405.68    821.01   4147.86   27428.45   23063    732.7     6.4
ARKANSAS             AR     1979    9554.83   4483.21    859.23   4212.40   28394.22   23501    749.4     6.2
```

Figure R3.7 Importing a Data Set that Contains Observation Tags

R3.2.3 Data Files that Contain Only Numeric Data

It is probably unlikely, but it is possible that your data file does not contain any variable names. For example, the data in a small file might appear as in Figure R3.8.

1	2	5
3	4	6
2	5	4
3	6	7

Figure R3.8 Small Data File

Use **IMPORT** exactly as before. The variables will be automatically named *x*1, *x*2, *x*3. This is likely to be a bit cumbersome – you may want to provide names for the variables. This can be done with the **READ** command as discussed in Section R3.5. However, a better solution would be to simply add the names to the data file and read in the names with **IMPORT**.

R3.2.4 Observation Labels without Variable Names in the Data File

Another possibility, probably also unlikely, is a data file with observation labels, but no variable names. Figure R3.9 shows the appearance.

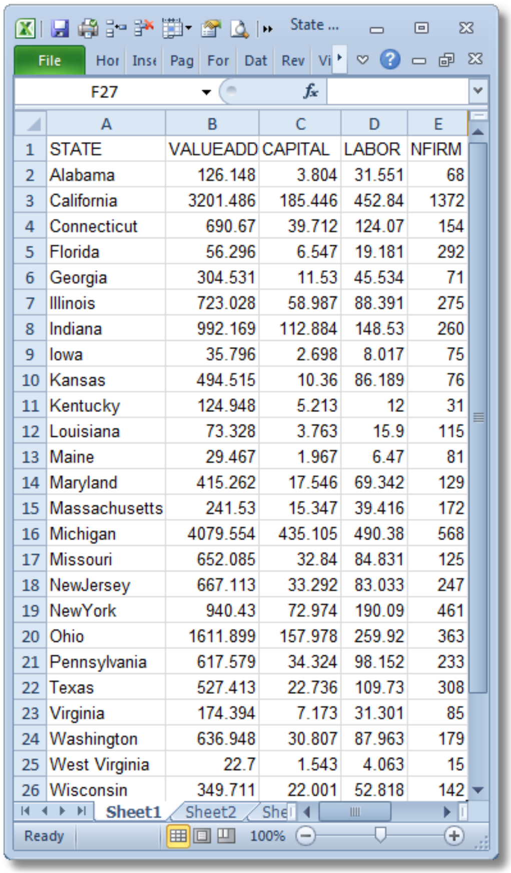
```
Alabama          126.148      3.804     31.551          68
California      3201.486    185.446    452.844        1372
Connecticut      690.670     39.712    124.074         154
Florida           56.296      6.547     19.181         292
Georgia          304.531     11.530     45.534          71
... (20 more observations)
```

Figure R3.9 Data File with Observation Labels and without Variable Names

You can use **IMPORT** exactly as before to read this data file. The variables will be named with the default names, *x*1, … To provide the variable names explicitly, use the **READ** command. Again, the preferable solution is to edit the file, add the variable names as a first row, and use **IMPORT.**

R3.2.5 Reading a Spreadsheet File from *Excel*

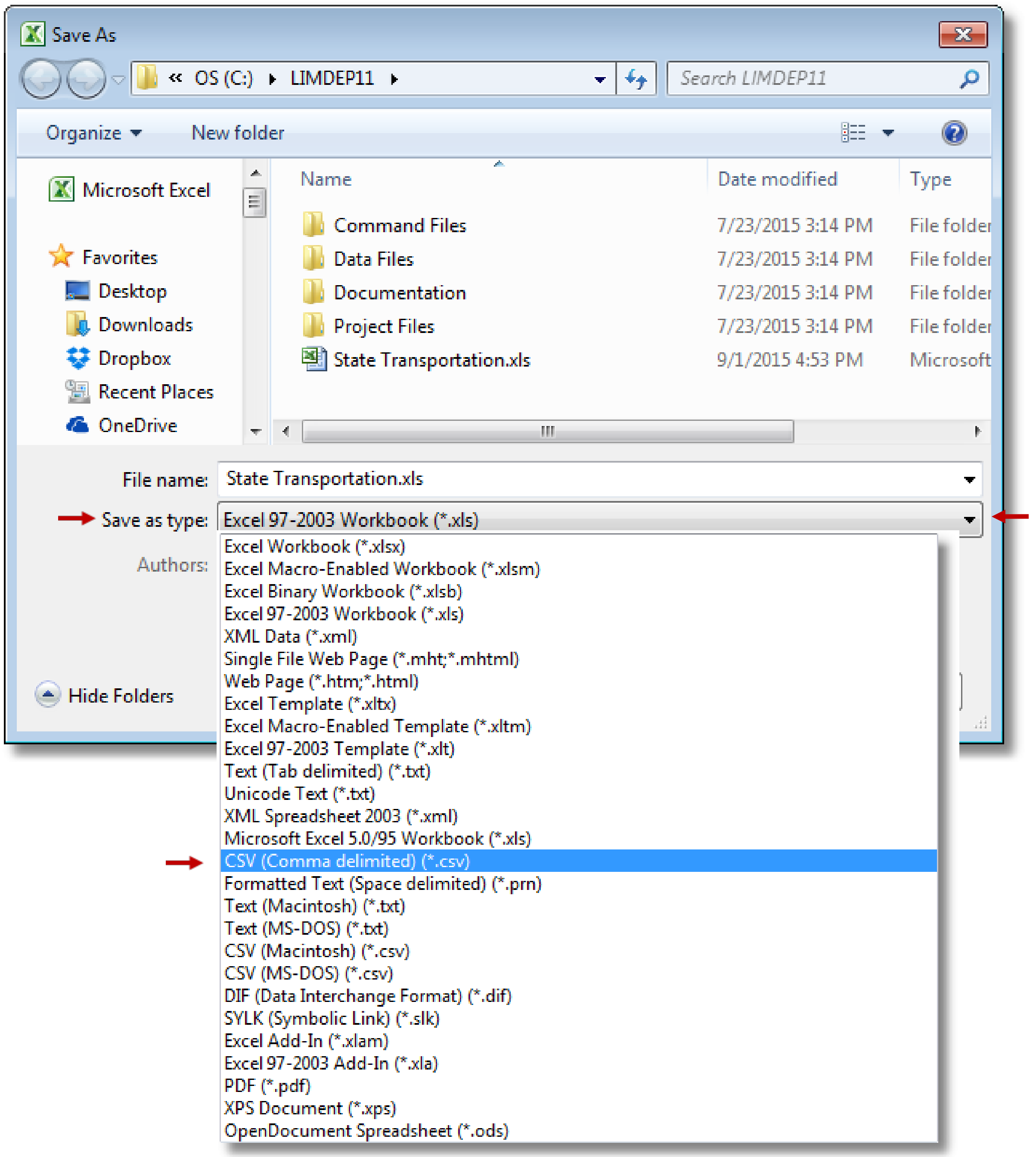
A sample data set read into *Excel* is shown in Figure R3.10. (This is Table F7.2 from Greene (2012), which contains 25 statewide observations on output and inputs in the transportation sector.) The data in this file are typical. In addition to the variable identifiers, the state names are part of the data set.



	A	B	C	D	E
1	STATE	VALUEADD	CAPITAL	LABOR	NFIRM
2	Alabama	126.148	3.804	31.551	68
3	California	3201.486	185.446	452.84	1372
4	Connecticut	690.67	39.712	124.07	154
5	Florida	56.296	6.547	19.181	292
6	Georgia	304.531	11.53	45.534	71
7	Illinois	723.028	58.987	88.391	275
8	Indiana	992.169	112.884	148.53	260
9	Iowa	35.796	2.698	8.017	75
10	Kansas	494.515	10.36	86.189	76
11	Kentucky	124.948	5.213	12	31
12	Louisiana	73.328	3.763	15.9	115
13	Maine	29.467	1.967	6.47	81
14	Maryland	415.262	17.546	69.342	129
15	Massachusetts	241.53	15.347	39.416	172
16	Michigan	4079.554	435.105	490.38	568
17	Missouri	652.085	32.84	84.831	125
18	NewJersey	667.113	33.292	83.033	247
19	NewYork	940.43	72.974	190.09	461
20	Ohio	1611.899	157.978	259.92	363
21	Pennsylvania	617.579	34.324	98.152	233
22	Texas	527.413	22.736	109.73	308
23	Virginia	174.394	7.173	31.301	85
24	Washington	636.948	30.807	87.963	179
25	West Virginia	22.7	1.543	4.063	15
26	Wisconsin	349.711	22.001	52.818	142

Figure R3.10 Sample *Excel* Data Set

You can import a file created by *Excel*. However, *LIMDEP* cannot read the default .xlsx format file that later versions of *Excel* (*Excel* 2007 and later) use to save a spreadsheet. First, you need to save the file in another file format. Select File (or the Microsoft Office button), then Save As. This will open a dialog box that lets you select an alternative file format, as shown in Figure R3.11. In the Save As dialog box, click the down arrow in the 'Save as type' window to view the file types. Then select the 'CSV (comma delimited)' format.


Save As
« OS (C:) ▸ LIMDEP11 ▸
Search LIMDEP11
Organize
New folder
Microsoft Excel
Favorites
Desktop
Downloads
Dropbox
Recent Places
OneDrive
Hide Folders
Authors:
Excel Workbook (*.xlsx)
Excel Macro-Enabled Workbook (*.xlsm)
Excel Binary Workbook (*.xlsb)
Excel 97-2003 Workbook (*.xls)
XML Data (*.xml)
Single File Web Page (*.mht;*.mhtml)
Web Page (*.htm;*.html)
Excel Template (*.xltx)
Excel Macro-Enabled Template (*.xltm)
Excel 97-2003 Template (*.xlt)
Text (Tab delimited) (*.txt)
Unicode Text (*.txt)
XML Spreadsheet 2003 (*.xml)
Microsoft Excel 5.0/95 Workbook (*.xls)
CSV (Comma delimited) (*.csv)
Formatted Text (Space delimited) (*.prn)
Text (Macintosh) (*.txt)
Text (MS-DOS) (*.txt)
CSV (Macintosh) (*.csv)
CSV (MS-DOS) (*.csv)
DIF (Data Interchange Format) (*.dif)
SYLK (Symbolic Link) (*.slk)
Excel Add-In (*.xlam)
Excel 97-2003 Add-In (*.xla)
PDF (*.pdf)
XPS Document (*.xps)
OpenDocument Spreadsheet (*.ods)

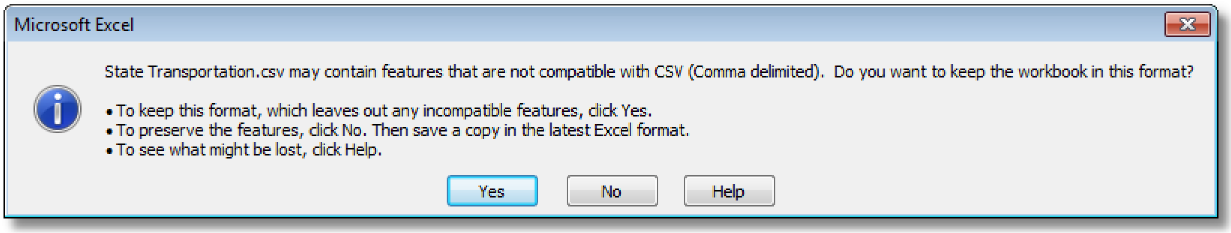
Name	Date modified	Type
Command Files	7/23/2015 3:14 PM	File folder
Data Files	7/23/2015 3:14 PM	File folder
Documentation	7/23/2015 3:14 PM	File folder
Project Files	7/23/2015 3:14 PM	File folder
State Transportation.xls	9/1/2015 4:53 PM	Microsoft

File name: State Transportation.xls

Save as type: Excel 97-2003 Workbook (*.xls)

Figure R3.11 Saving an *Excel* File in CSV format

Excel presents a warning dialog box that the file may contain features that are not compatible with the CSV format. Just click Yes to proceed.



State Transportation.csv may contain features that are not compatible with CSV (Comma delimited). Do you want to keep the workbook in this format?

- To keep this format, which leaves out any incompatible features, click Yes.
- To preserve the features, click No. Then save a copy in the latest Excel format.
- To see what might be lost, click Help.

Figure R3.12 Warning Dialog Box in CSV format

NOTE ON FILE FORMATS: *LIMDEP* can read .xls files written by *Excel* 2003. This is one of the formats available when you use Save As in *Excel*. However, an *Excel* 2007 or later .xlsx file saved as an *Excel* 97-2003 .xls file cannot be read by *LIMDEP*. The .xls file created by this choice is compatible with earlier versions of *Excel*, but usually not with other software. If you have an existing .xls or .xlsx file that contains your data for export to *LIMDEP*, open the file in *Excel* and use Save As to save the file in the .csv format. You should always use CSV files to export data from *Excel* to *LIMDEP*.

If your *Excel* file follows the standard format described in Section R3.2, you can simply read in the .csv file using the Import menu option or the **IMPORT** command. (Be sure to remove any commas in the cells and any spaces in the variable names.) If your *Excel* file does not follow the standard format, then you may use the **READ** command to input the file, described in Section R3.5.

NOTE ON SPREADSHEET DIMENSIONS: An advantage of the generic format is that it has relaxed the constraints on spreadsheet sizes that were built into the .xls format. In versions of *Excel* before 2007, the limits were 255 columns and 65536 rows. The new limits are 65536 columns and 1,048,576 rows.

NOTE ON FORMULAS IN CELLS: Another advantage of the CSV format is that it is not affected by whether the cells in your spreadsheet contain values or formulas. The item that *Excel* puts in the CSV file will always be the number that you see on your screen even if that number is placed there by a formula in the background. You don't have to worry about formulas vs. values in your spreadsheet file.

TIP: If you regularly use other programs to create data sets to transport to *LIMDEP* or *NLOGIT*, including *Excel*, *SAS*, *Stata*, *SPSS*, or vice versa, you will find the utility program *StatTransfer* a worthwhile acquisition. *StatTransfer* is discussed in Section R3.5.6.

R3.2.6 Missing Values in Data Files

LIMDEP will catch nonnumeric or missing data codes in most types of data sets. In general, any value not readable *as a number* is considered a missing value and given the value -999.

In all settings, -999 is *LIMDEP*'s internal missing data code.

Since -999 is a distinctive, but otherwise legitimate value, no account is taken of missing data in estimation. It is up to you to **REJECT** observations for which the missing value has been inserted. (**REJECT** is discussed in Chapter R7. A convenient, global means of handling missing data is also discussed in Chapter R7.) Some things to remember about missing data are:

- A blank in a data file is normally not a missing value; it is just a blank. Since a data file imported or read without a format can be arranged any way you want, *LIMDEP* has no way of knowing that a blank is supposed to be interpreted as a missing value. But, all other nonnumeric, nonblank entries are treated as missing. This includes the familiar '.' character, the word 'missing,' or any other code you care to use. A method is provided in Section R3.5.7 for you to tell *LIMDEP* to treat blank fields as missing data.

- There will be occasions when *LIMDEP* claims it found missing values when you did not think there were any. The cause is usually an error in a **READ** command or some other problem in the file. For example, you can provide data in a file on more than one line. But, you must not end a line with a comma. This particular error will lead to missing values showing up unexpectedly.

R3.2.7 Missing Values in the Comma Delimited (CSV) Files

IMPORT is used for reading any text file that has comma, tab, or space separated values. This includes CSV files, text files, or any other ASCII formatted file. The flexibility of this arrangement makes it impossible to determine that a space is meant to indicate a missing value. However, the CSV format in particular (and not other types of files) has a distinctive, easily detectable method of indicating missing values. In a CSV file, missing values, and only missing values, are shown as blanks. This means that when you specifically tell *LIMDEP* that your data file is in the CSV format, it can find the missing values by locating the blanks in the data. In a CSV file, missing values at the beginning, middle, and end of a line of data are indicated by 'blank then comma,' 'comma, then blank, then comma,' and 'comma then blank.' Because of the last of these, you must be sure that you never use a comma at the end of a line in a CSV file unless you intend for that comma to be followed by a missing value. Programs that create this type of file will respect this convention, so the potential error will only arise if you yourself manipulate the contents of the file. This special case is handled with the **READ** command. See Section R3.5.

R3.2.8 Changing the Missing Value Code

The default missing value code in *LIMDEP* is -999. Throughout this manual, we will assume this convention. On reading a data file, when a nonnumeric value is encountered, it is replaced with the default missing value. Other operations, such as transformations, will work around the missing values, as described in the preceding section and elsewhere. You can change the default missing value if necessary. (Do this with caution.) There are two ways to do so. First, use

CALC ; MissingValue = the value $

For example,

CALC ; MissingValue = -888 $

After this command, the program will respond

```
+----------------------------------------------------+
| NOTE! Default missing value has been changed from |
| Old:  -999.0000000     to New:    -888.0000000     |
| Value may be reset with CALC;MissingValue=value $ |
+----------------------------------------------------+
```

A new scalar named *dfltmsng* will appear in the project window, and you can verify that the value has now been changed. Do note, that after making this change, -999 would become a valid numeric result, no longer a missing value. Important points to remember

- The default missing value is a setting for the current data set, not a program setting. After you exit the program and restart, the default missing value will revert to -999.
- The default missing value is retained in the project file if you save the project after you change the default missing value. Thus, in the example above, if we now save the project file and exit, then restart the program, the default missing value will again be -999. If we then reload the project file, the value will be reset to -888, as it was when the project was saved.

You can also change the default missing value in the data file itself. The **READ** or **IMPORT** command would be modified as follows:

IMPORT (or READ) ; MissingValue = the value ; … the rest of the command $

When the file is read, nonnumeric data (and the value given) become missing values. (Note, -999 would now become a valid value.)

R3.2.9 Data Files that Are Not Formatted for IMPORT

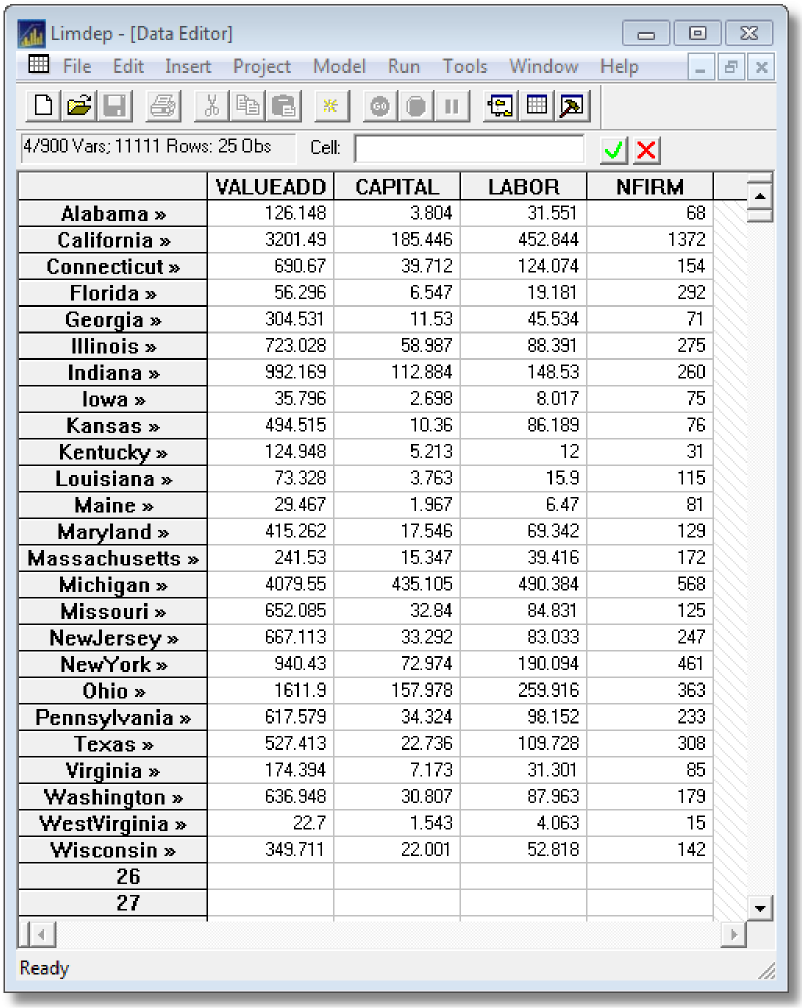
Since you do not provide any information with the **IMPORT** command except for the name of the file, *LIMDEP* must try to determine from the file itself what the appropriate layout is, that is, whether there are labels and/or names in the file, how many variables there are, and so on. In most cases, this will be transparent, and your file will be imported the way you expected. If you find that the imported data have missing values where you did not expect them, or if variables seem to be in the wrong columns, then the file is probably not arranged appropriately for **IMPORT**.(You can use the data editor described in the next section to view your data.) In this case, you should reset the project (with Project:Reset), and try again to read the data, this time with the **READ** command, which is discussed in Section R3.5.

Note, finally, there are cases in which the data are explicitly not arranged the way **IMPORT** would expect them. For example, if you have observation labels in the file and the numeric data are given on more than one line, then the data will not be imported properly. In these cases, you must use **READ**, not **IMPORT**. (If you do not have observation labels, then **IMPORT** can handle data on more than one line. However, with **IMPORT**, if the first row is variable names, then regardless of how the numeric data are arranged, the variable names must appear on a single line.)

R3.3 The Data Editor

LIMDEP's data editor contains data that you import or read and any new variables that you create. The data editor resembles familiar spreadsheet programs, such as Microsoft *Excel*. You can reach the data editor in several ways:

- Click the data editor (grid) icon on the *LIMDEP* toolbar.
- Double click any variable name in the project window.
- Select the menu option Project:Data Editor.



4/900 Vars; 11111 Rows: 25 Obs Cell:

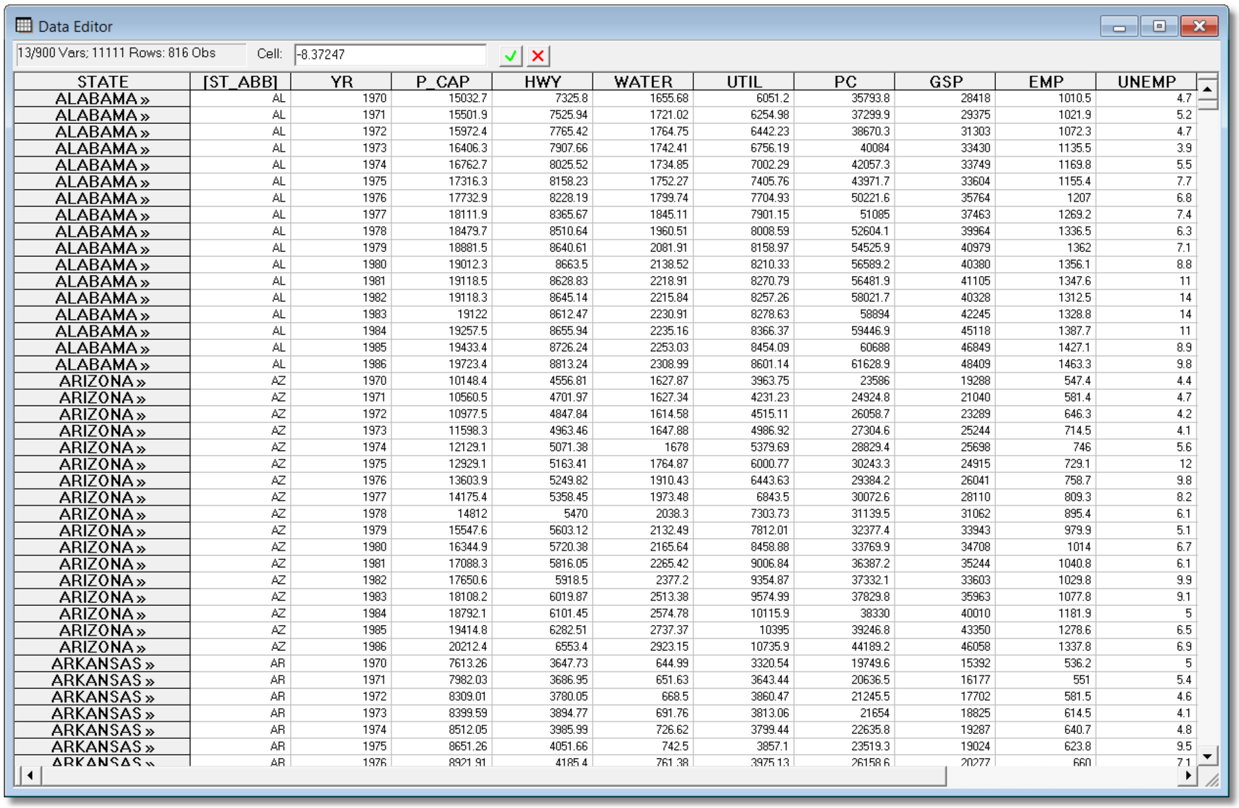
	VALUEADD	CAPITAL	LABOR	NFIRM
Alabama »	126.148	3.804	31.551	68
California »	3201.49	185.446	452.844	1372
Connecticut »	690.67	39.712	124.074	154
Florida »	56.296	6.547	19.181	292
Georgia »	304.531	11.53	45.534	71
Illinois »	723.028	58.987	88.391	275
Indiana »	992.169	112.884	148.53	260
Iowa »	35.796	2.698	8.017	75
Kansas »	494.515	10.36	86.189	76
Kentucky »	124.948	5.213	12	31
Louisiana »	73.328	3.763	15.9	115
Maine »	29.467	1.967	6.47	81
Maryland »	415.262	17.546	69.342	129
Massachusetts »	241.53	15.347	39.416	172
Michigan »	4079.55	435.105	490.384	568
Missouri »	652.085	32.84	84.831	125
NewJersey »	667.113	33.292	83.033	247
NewYork »	940.43	72.974	190.094	461
Ohio »	1611.9	157.978	259.916	363
Pennsylvania »	617.579	34.324	98.152	233
Texas »	527.413	22.736	109.728	308
Virginia »	174.394	7.173	31.301	85
Washington »	636.948	30.807	87.963	179
WestVirginia »	22.7	1.543	4.063	15
Wisconsin »	349.711	22.001	52.818	142
26				
27				

Figure R3.13 Active Data Editor

The display shows you precisely what appears in the data array. The chevrons (››) next to the observation numbers or labels indicate that these observations are in the current sample. See Chapter R7.

NOTE: The data editor can only display 5,000 rows, regardless of how many actual rows of data you have. If you have read more than 5,000 observations, and you go to the data editor, it will still only show 5,000 rows. This does not mean that your data were not read completely. To verify the import of a data set, you can use **CALC ; List ; N $** or **DSTAT ; Rhs = * $** to show you the actual observation count. The current sample length will also appear at the top of the project window at all times. The **CALC** command is discussed in Chapter R17.

If your data contains observation tags, they will appear in the data editor as a variable, as shown in Figure R3.14.



13/900 Vars; 11111 Rows: 816 Obs

Cell: -8.37247

STATE	[ST_ABB]	YR	P_CAP	HWY	WATER	UTIL	PC	GSP	EMP	UNEMP
ALABAMA »	AL	1970	15032.7	7325.8	1655.68	6051.2	35793.8	28418	1010.5	4.7
ALABAMA »	AL	1971	15501.9	7525.94	1721.02	6254.98	37299.9	29375	1021.9	5.2
ALABAMA »	AL	1972	15972.4	7765.42	1764.75	6442.23	38670.3	31303	1072.3	4.7
ALABAMA »	AL	1973	16406.3	7907.66	1742.41	6756.19	40084	33430	1135.5	3.9
ALABAMA »	AL	1974	16762.7	8025.52	1734.85	7002.29	42057.3	33749	1169.8	5.5
ALABAMA »	AL	1975	17316.3	8158.23	1752.27	7405.76	43971.7	33604	1155.4	7.7
ALABAMA »	AL	1976	17732.9	8228.19	1799.74	7704.93	50221.6	35764	1207	6.8
ALABAMA »	AL	1977	18111.9	8365.67	1845.11	7901.15	51085	37463	1269.2	7.4
ALABAMA »	AL	1978	18479.7	8510.64	1960.51	8008.59	52604.1	39964	1336.5	6.3
ALABAMA »	AL	1979	18881.5	8640.61	2081.91	8158.97	54525.9	40979	1362	7.1
ALABAMA »	AL	1980	19012.3	8663.5	2138.52	8210.33	56589.2	40380	1356.1	8.8
ALABAMA »	AL	1981	19118.5	8628.83	2218.91	8270.79	56481.9	41105	1347.6	11
ALABAMA »	AL	1982	19118.3	8645.14	2215.84	8257.26	58021.7	40328	1312.5	14
ALABAMA »	AL	1983	19122	8612.47	2230.91	8278.63	58894	42245	1328.8	14
ALABAMA »	AL	1984	19257.5	8655.94	2235.16	8366.37	59446.9	45118	1387.7	11
ALABAMA »	AL	1985	19433.4	8726.24	2253.03	8454.09	60688	46849	1427.1	8.9
ALABAMA »	AL	1986	19723.4	8813.24	2308.99	8601.14	61628.9	48409	1463.3	9.8
ARIZONA »	AZ	1970	10148.4	4556.81	1627.87	3963.75	23586	19288	547.4	4.4
ARIZONA »	AZ	1971	10560.5	4701.97	1627.34	4231.23	24924.8	21040	581.4	4.7
ARIZONA »	AZ	1972	10977.5	4847.84	1614.58	4515.11	26058.7	23289	646.3	4.2
ARIZONA »	AZ	1973	11598.3	4963.46	1647.88	4986.92	27304.6	25244	714.5	4.1
ARIZONA »	AZ	1974	12129.1	5071.38	1678	5379.69	28829.4	25698	746	5.6
ARIZONA »	AZ	1975	12929.1	5163.41	1764.87	6000.77	30243.3	24915	729.1	12
ARIZONA »	AZ	1976	13603.9	5249.82	1910.43	6443.63	29384.2	26041	758.7	9.8
ARIZONA »	AZ	1977	14175.4	5358.45	1973.48	6843.5	30072.6	28110	809.3	8.2
ARIZONA »	AZ	1978	14812	5470	2038.3	7303.73	31139.5	31062	895.4	6.1
ARIZONA »	AZ	1979	15547.6	5603.12	2132.49	7812.01	32377.4	33943	979.9	5.1
ARIZONA »	AZ	1980	16344.9	5720.38	2165.64	8458.88	33769.9	34708	1014	6.7
ARIZONA »	AZ	1981	17088.3	5816.05	2265.42	9006.84	36387.2	35244	1040.8	6.1
ARIZONA »	AZ	1982	17650.6	5918.5	2377.2	9354.87	37332.1	33603	1029.8	9.9
ARIZONA »	AZ	1983	18108.2	6019.87	2513.38	9574.99	37829.8	35963	1077.8	9.1
ARIZONA »	AZ	1984	18792.1	6101.45	2574.78	10115.9	38330	40010	1181.9	5
ARIZONA »	AZ	1985	19414.8	6282.51	2737.37	10395	39246.8	43350	1278.6	6.5
ARIZONA »	AZ	1986	20212.4	6553.4	2923.15	10735.9	44189.2	46058	1337.8	6.9
ARKANSAS »	AR	1970	7613.26	3647.73	644.99	3320.54	19749.6	15392	536.2	5
ARKANSAS »	AR	1971	7982.03	3686.95	651.63	3643.44	20636.5	16177	551	5.4
ARKANSAS »	AR	1972	8309.01	3780.05	668.5	3860.47	21245.5	17702	581.5	4.6
ARKANSAS »	AR	1973	8399.59	3894.77	691.76	3813.06	21654	18825	614.5	4.1
ARKANSAS »	AR	1974	8512.05	3985.99	726.62	3799.44	22635.8	19287	640.7	4.8
ARKANSAS »	AR	1975	8651.26	4051.66	742.5	3857.1	23519.3	19024	623.8	9.5
ARKANSAS »	AR	1976	8921.91	4185.4	761.38	3975.13	26158.6	20277	660	7.1

Figure R3.14 Data Editor with Observation Tags

Note that the tags appear with the rest of the variables in the data set. Tags are identified in the data editor and in operations that use them by enclosing the name in square brackets, as can be seen in the figure.

You need not have read a data set to access the data editor. The data editor appears as shown in Figure R3.15, with an empty editing area when no variables exist.

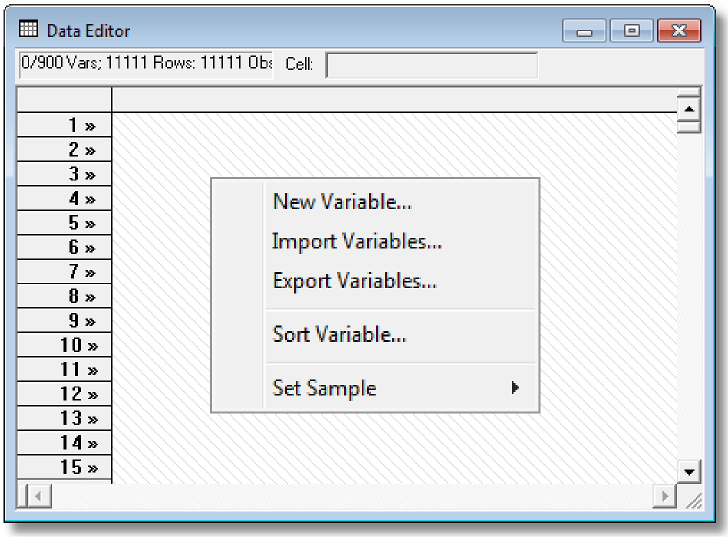
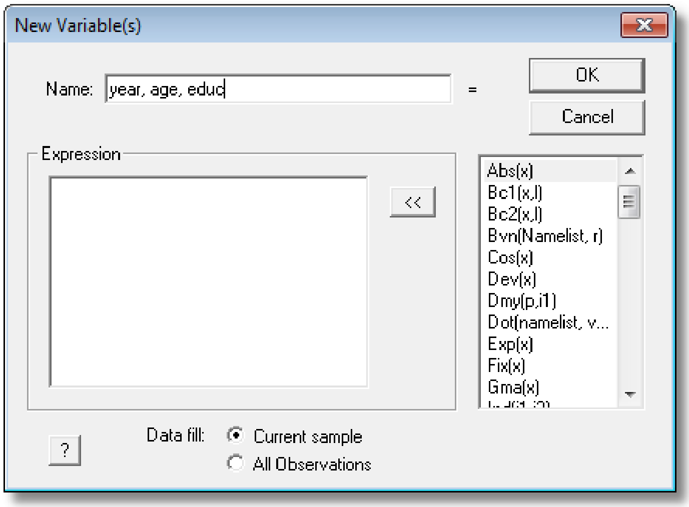


Figure R3.15 Data Editor and 'Right Mouse' Menu

The functions of the data editor are shown in the smaller menu, which you obtain by clicking the *right* mouse button, as displayed in Figure R3.15. The functions of the data editor menu are described below.

New Variable

You can enter a small data set by typing in the data in the editor. First, it is necessary to create the columns. To do so, select New Variable in the menu to open the dialog box shown in Figure R3.16.


New Variable(s)

Name: year, age, educ

Data fill: [✓ Current sample] All Observations

Figure R3.16 Entering New Variables in New Variable Dialog Box

Just enter the names of the new variables in the window and click OK. For our example, the variables names are *year*, *age*, *educ*. Variable names may have up to eight characters, must begin with a letter, and must be composed from only letters, numbers and the underscore character. Variable names will be converted to upper case with spaces removed. After clicking OK, the window changes to that in Figure R3.17, and you can begin typing in the data.

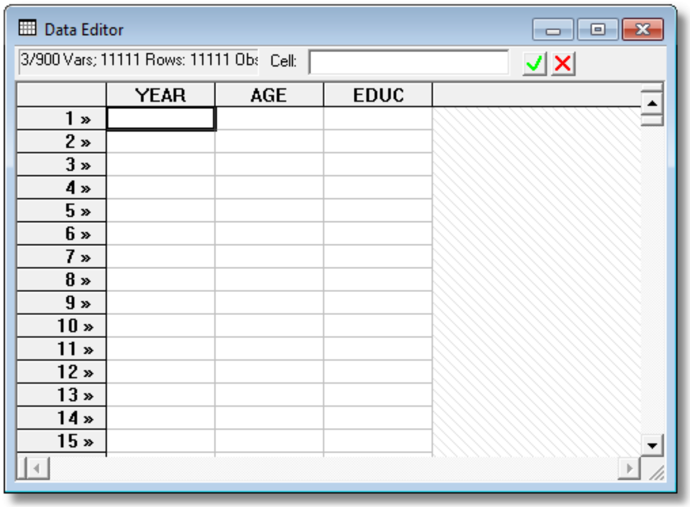
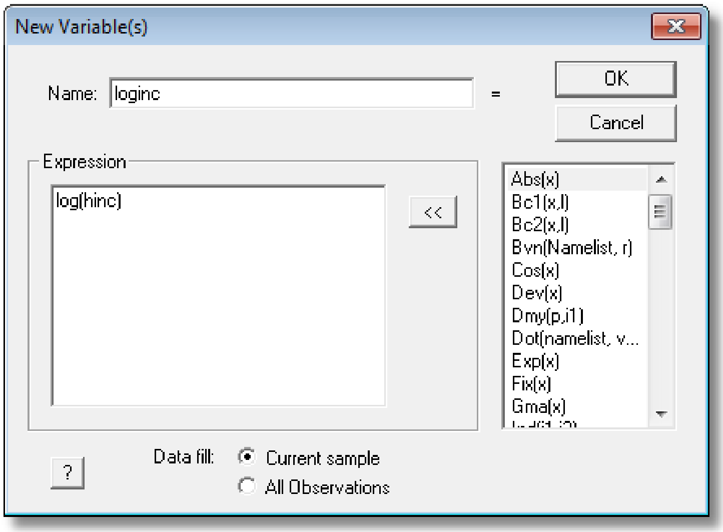


Figure R3.17 Data Editing Window with New Variables

NOTE: If your data have missing values, they will appear as blank cells in the data editor. Although a missing value is displayed as a blank, internally, the cell contains -999 (or the default value you have defined for missing – see Section R3.2.6).

HINT: The data editor does not reset the current sample. After you enter a data set with it, the current sample is unchanged. If you enter a data set initially in the data editor, you should use Set Sample or a **SAMPLE** command to set the sample appropriately. (See Chapter R7.)

When there are data already stored in the data area, the New Variable dialog box can be used to compute transformed variables. This is equivalent to the **CREATE** command described in Chapter R4. For example, in Figure R3.18, the dialog box is being used to create a variable named *loginc* using the existing variable named *hinc* and the built-in log function.



Name: loginc =

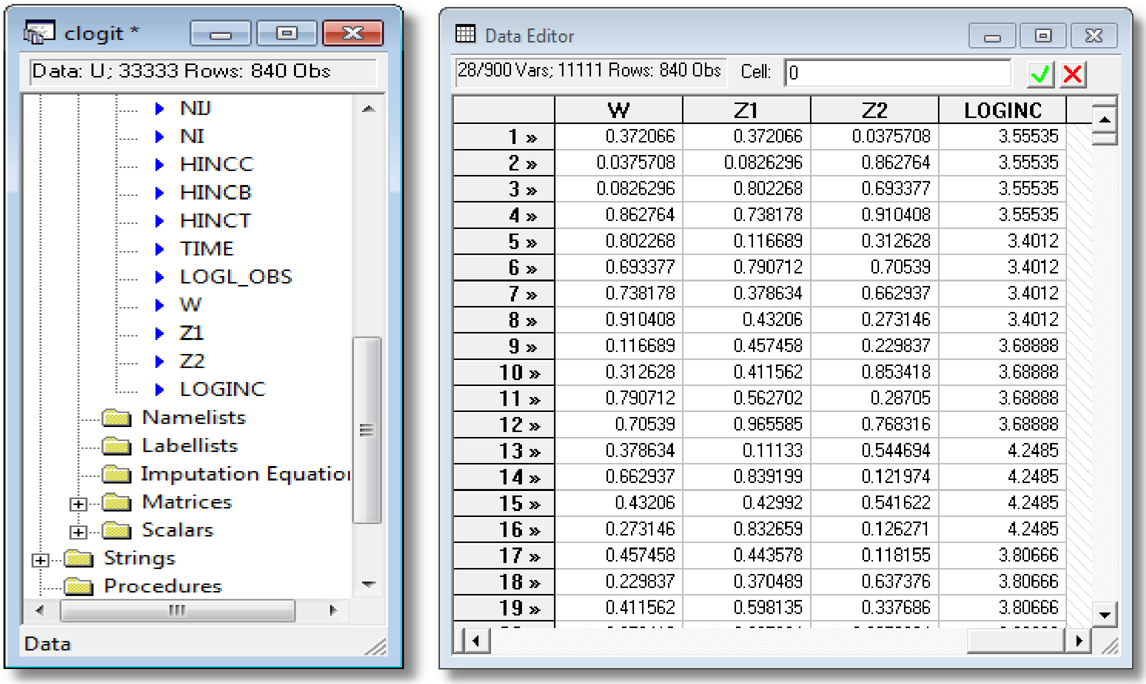
log(hinc)

Data fill: [✓ Current sample] All Observations

Figure R3.18 Transforming Variables in New Variable Dialog Box

After you create and enter new variables, the project window is updated, and the data, themselves, are placed in the data area. Figure R3.19 illustrates.

HINT: The New Variable dialog box will allow you to replace the existing values of a variable. For example, in Figure R3.18, if '*loginc*' were '*hinc*' in the Name window, then the values of *hinc* would simply be replaced with their logarithms.



	W	Z1	Z2	LOGINC
1 »	0.372066	0.372066	0.0375708	3.55535
2 »	0.0375708	0.0826296	0.862764	3.55535
3 »	0.0826296	0.802268	0.693377	3.55535
4 »	0.862764	0.738178	0.910408	3.55535
5 »	0.802268	0.116689	0.312628	3.4012
6 »	0.693377	0.790712	0.70539	3.4012
7 »	0.738178	0.378634	0.662937	3.4012
8 »	0.910408	0.43206	0.273146	3.4012
9 »	0.116689	0.457458	0.229837	3.68888
10 »	0.312628	0.411562	0.853418	3.68888
11 »	0.790712	0.562702	0.28705	3.68888
12 »	0.70539	0.965585	0.768316	3.68888
13 »	0.378634	0.11133	0.544694	4.2485
14 »	0.662937	0.839199	0.121974	4.2485
15 »	0.43206	0.42992	0.541622	4.2485
16 »	0.273146	0.832659	0.126271	4.2485
17 »	0.457458	0.443578	0.118155	3.80666
18 »	0.229837	0.370489	0.637376	3.80666
19 »	0.411562	0.598135	0.337686	3.80666

Figure R3.19 Data Editor and Project Window

Import Variables

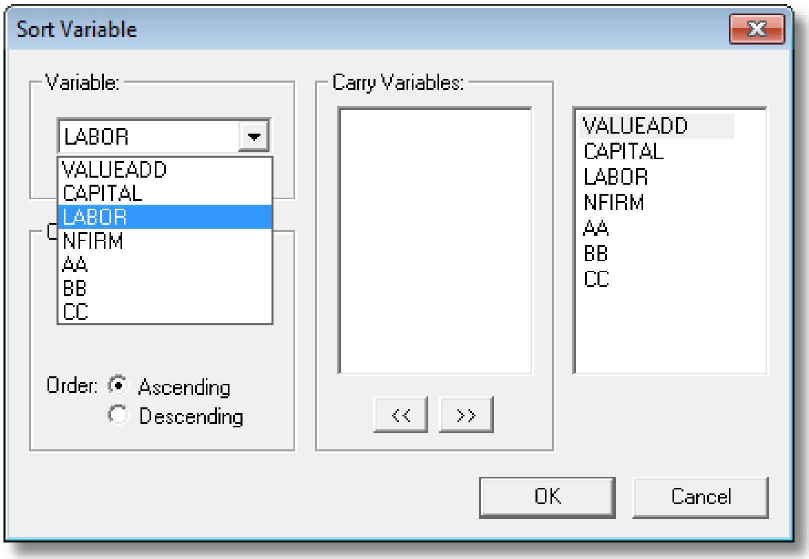
Click Import Variables to open a dialog box that allows you to import a spreadsheet or other data file. This is the same as selecting Project:Import/Variables or the **IMPORT** command described in Section R3.2.

Export Variables

Click Export Variables to open a dialog box that allows you to write variables in a data file. You can create a.csv file. This is the same as the **EXPORT** command. **EXPORT** and an extension, **WRITE**, are discussed in Section R3.9.

Sort Variable

Click Sort Variable to open a dialog box that allows you to sort a variable carrying other variables with it (optionally). The dialog box for this operation, which is the same as the **SORT** command (see Chapter R4) is shown in Figure R3.20.



Order: [✓ Ascending] Descending

Figure R3.20 Dialog Box for Sort Variable

Set Sample

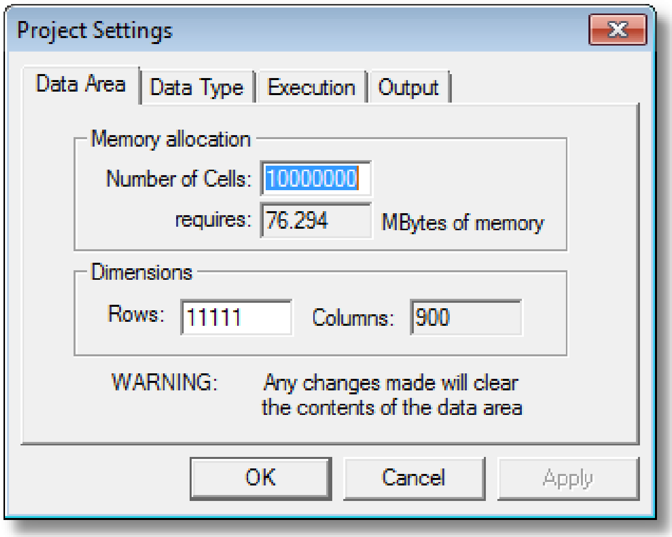
Click Set Sample to obtain a menu of options for setting the current sample. This uses the All, Range, Include, Reject and Draw options discussed in Chapter R7 to set the current sample. (This option is not available until data have been read into the data area.)

R3.4 The Data Area

Your data are stored in an area of memory that we will refer to as the data array. The data area is originally configured with 5,000,000 cells. This should be sufficient for most applications. However, the program allows you to analyze at least 3,000,000 observations (or more). It is conceivable that your data set might be too large for the original setting. If so, you need to reset the size of the data area. This is not done automatically because although the program can figure out how many cells you need to import your data, it cannot guess how many more cells you need for new variables you might create. This section shows you how to adjust the data area.

R3.4.1 Temporary Expansion of the Data Area

To see the current size of the data array, select Project:Settings, then click the Data Area tab, as shown in Figure R3.21. This dialog box allows you to adjust the number of cells in the data area for the session you are currently using. The memory requirement displayed is the number of megabytes, determined as (8×number of cells)/(1024×1024). (Each number requires eight bytes.) Thus, the 76.294 megabytes required is 80 million divided by 1024^2.



Memory allocation

Number of Cells: 10000000

requires: 76.294 MBytes of memory

Dimensions

Rows: 11111 Columns: 900

WARNING: Any changes made will clear the contents of the data area

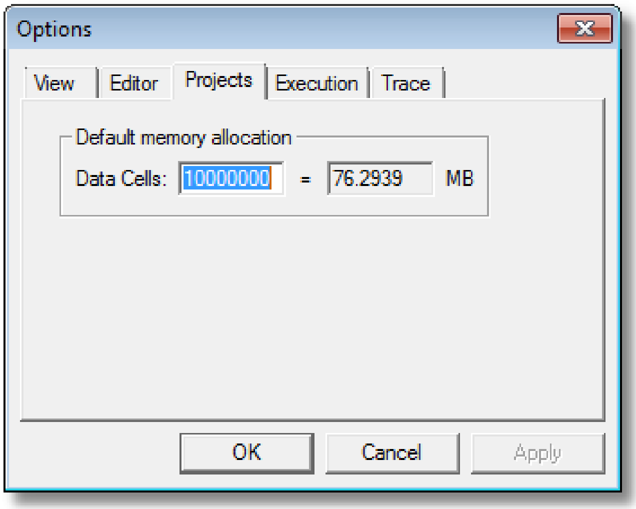
Figure R3.21 Project:Settings/Data Area

TIP: Do not use this feature after you import or read a data set. In order to expand the data area, it must be cleared of all existing data. Use this feature before you import or read your data.

As noted, this setting is only for the current session. When you exit *LIMDEP* and come back later, the data area will once again be set to what it was before you made this change (probably 5,000,000 cells).

R3.4.2 Permanently Setting the Number of Cells in the Data Area

You can change the size of the data area permanently (until you change it again) by using Tools:Options/Projects to change the default setting. The dialog box shown in Figure R3.22 shows how to make this setting. You may change this at any time.



Default memory allocation

Data Cells: 10000000 = 76.2939 MB

Figure R3.22 Tools:Options/Projects

Modern computers generally have plenty of memory. You will probably find a permanent setting such as 10,000,000 to be sufficient for your needs without placing excessive demands on your system.

R3.4.3 Setting the Number of Rows in the Data Area

The data array is initially configured as a rectangle with 900 columns and NKMAX/900 rows where NKMAX is the number of cells in your data area. When you enter data, they are placed at the top of this array moving down the rows, in the natural fashion. When you read a data set, if you require more than the default number of rows, the data array is automatically reconfigured with more rows and fewer columns if necessary. However, if you are using the random number generators to produce random samples, rather than analyzing an externally produced data set, you may wish to change the number of rows yourself to allow you to have a larger number of observations. To change the number of rows in the data area, use the Project:Settings/Data Area dialog box shown in Figure R3.21. You will be reminded when you do this that changing the configuration of your data area erases all existing data. Note, the dialog box in Figure R3.21 will not allow you to reduce the number of rows from its current setting. You can use the **SAMPLE** command to do that.

You can also reconfigure the data array (without resizing it) by using the **ROWS** command:

ROWS ; the value $

(If you give the unmodified command, **ROWS $**, you will be offered the dialog box in Figure R3.23 in which you can provide the desired number of rows.) When you use this command, existing data are erased.

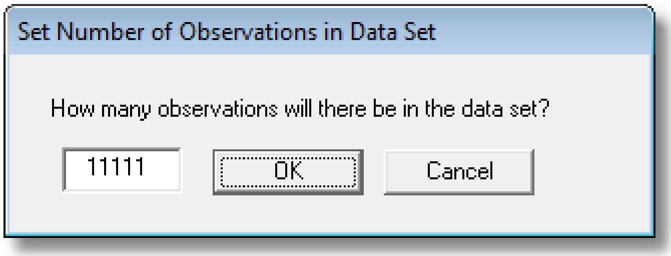

Set Number of Observations in Data Set
How many observations will there be in the data set?
11111
OK
Cancel

Figure R3.23 ROWS Dialog Box

HINT: If you are reading more than one data file, read the longest one first. Once the data area has been reconfigured to accommodate a data set, it cannot be reconfigured again without losing the first data set.

NOTE: You are limited to 900 variables in your active data set when you operate *LIMDEP*. The 900 variable limit is always effective so you cannot reduce the default number of rows.

R3.5 The READ Command for Nonstandard Data Files

The **IMPORT** command shown in Section R3.2 will succeed for most types of data files that you will use. But, there are many kinds of data files, and many ways for data files to differ from the standard format assumed for **IMPORT**. This section will describe different strategies for importing your data. The general command for reading a data file is that is not read properly by **IMPORT** is

```
READ        ; Nvar   = number of variables
            ; Nobs   = number of observations
            ; Names  = list of names for the variables
            ; File   = the full file name, including path
            ; Format = identifier for certain types of files
            ; Labels = position in the file where labels are found
            ; By Variables for a certain arrangement of data $
```

Several of the specifications listed are optional and are used only for specific situations. The **READ** command is submitted from the text editing window, just like the **IMPORT** command. It is helpful to use Insert:File Path to obtain the full path for the file name.

R3.5.1 ASCII Numeric Data Files

This assumes that the file is an ASCII file (not binary, not a spreadsheet, etc.) with numbers arranged in rows, separated by blanks and/or commas. Numbers in the file need not be neatly arranged vertically. If there are missing values, there must be placeholders for them since blanks just separate values, they cannot be interpreted as missing data. Use as many lines as needed for each observation to supply all of the values.

If the data appear on a single line for each observation, then you can use **IMPORT** to read the data file. That case is shown in Figure R3.8. The variables will be automatically named *x*1,*x*2,… In order to provide the names, you must use **READ** as described in this section.

The situation considered here is that in which the data in the file appear on more than one line per observation. The data in Figure R3.24 are an example – there are four observations on three variables.

```
1
2  5
3  4  6
2  5  4
3  6
7
```

Figure R3.24 Data on Multiple Lines

The reason this file cannot be read with **IMPORT** is that there is no way to determine how many variables it contains, since observations can be on more than one line. This would be true even if each observation were on a single line, since it would be valid for it to have two observations on six variables rather than four on two.

TIP: IMPORT is clever. The obstacle is the first line. If your data set has a full set of observations on the first line, but later observations take more than one line, then **IMPORT** will read the file correctly if the numeric data are in the expected format (comma or space delimited), since it will guess correctly that the number of variables in the file is the number that appear on the first line. This section is about files that generally require more than one line per observation, including the first observation.

To read this file, you can use **READ** with

```
READ          ; File = filename ; Nobs = 4 ; Nvar = 3 $
```

If you omit the names, the variables will be named *x*1, *x*2, *x*3. You must generally provide **; Nobs** and **; Nvar**.

TIP: If this file had three names on a single row in the first line, **IMPORT** would read the file correctly if the observations to follow take the usual format (comma or space delimited). The names would suffice to identify the shape of the data file.

HINT: If you do not know the exact number of observations in your data set, give **; Nobs** a number that you are sure will be larger than the actual value. *LIMDEP* will just read to the bottom of the file and adjust the number of observations appropriately.

R3.5.2 Variable Names Not Provided in the Data File

The most convenient way to supply variable names is in the first line of the data file, itself. This is the common approach used by **IMPORT**. But, if your data file contains only the data, and not the names, then you can provide them.

The normal way to enter variable names is in the command.

```
              ; Names = name_1,...,name_nvar
```

The variables in the file in Figure R3.24 can be named *bill*, *jim*, *bob* with

```
READ          ; File = filename ; Nobs = 4 ; Nvar = 3 ; Names = bill,jim,bob $
```

Variable names may have up to eight characters, must begin with a letter, and must be composed from only letters, numbers, and the underscore character. Remember that names are always converted to upper case. Reserved names are listed in Section R2.6.2.

R3.5.3 Variable Names in the Data File

The more convenient way to provide names is in the data file. If this file had three names *on a single row* in the first line, **IMPORT** will read the file correctly. However, if the variable names are on more than one line, you cannot use the **IMPORT** command. Figure R3.25 continues the earlier example

```
bill, jim,
bob
1
2  5
3  4  6
2  5  4
3  6
7
```

Figure R3.25 Variable Names Provided with Data on Multiple Lines

In this case, the variable names are on two lines. To indicate that the variable names are in the file, use

; Names = n

in your **READ** command, where *n* is the number of lines you need to list the names. Then, at the absolute beginning of the data file, include exactly *n* lines containing the variable names, separated by any number of spaces and/or commas. The command to read the data set in Figure R3.25 would be

READ ; File = filename ; Nvar = 3 ; Nobs = 4 ; Names = 2 $

R3.5.4 Observation Labels

IMPORT can read a data file with the observations labels in the first column only. If your data file has observation labels in any other column, use the following format to read your file:

READ ; File = filename ; Nobs = ... ; Nvar = ...
; Labels = the column in the data file that contains the labels $

NOTE: If your observation labels are not in the first column and your data file has variable names, you must also include **; Names = n**, even if the variables names are on a single line, which you would indicate with **; Names = 1**.

The data file in Figure R3.26 contains labels for the individual observations in the second column, as well as variable names.

```
ValueAdd     State          Capital     Labor     NFirm
126.148      Alabama          3.804    31.551        68
3201.486     California     185.446   452.844      1372
690.670      Connecticut     39.712   124.074       154
56.296       Florida          6.547    19.181       292
304.531      Georgia         11.530    45.534        71
... (20 more observations)
```

Figure R3.26 Data File with Observation Labels

To read the file in Figure R3.26, you would use

READ ; File = … ; Nvar = 4 ; Nobs = 25
; Names = 1 ; Labels = 2 $

The following rules apply to the syntax:

- The maximum number of observations in a labeled file is 65,536.
- The labels column is an extra column in the data. It is not a variable.
- Nvar does not include the labels.
- You must include a name for the labels. Note that the name 'State' is used for the labels, but not for the data.
- Data may not be 'transposed.' (See Section R3.5.5.)
- The file may not be formatted. (See Section R3.5.7.)

The situation that this form of **READ** is provided for is that in which the observation labels are not in the first column (or the data are nonstandard formatted).

R3.5.5 Transposed Data Files – Reading by Variables

There are two ways to arrange a data set, 'by observation' and 'by variable.' When data are read by observation, each line (or, possibly, group of lines) is a single observation, perhaps a year or individual, on one or more variables. This is how the preceding examples have been arranged. When you enter data 'by variables,' you will provide the full set of observations on a variable, then proceed to the next variable, and so on.

You may find it more convenient to enter data one variable at a time instead of one row at a time. If your data are arranged by variables, instead of by observations, you can **READ** them in transposed form just by adding

; By Variables

to the **READ** command. If you use this option, you must also include an accurate value for

; Nobs = number of observations

This is no longer optional. Also, if you specify **; By Variables**, you must include **; Names = n**, even if the names are on a single line. For example, two ways to **READ** the following data matrix are: (The names are not part of either data set.)

Arranged by Observations		
(year)	**(gdp)**	**(cons)**
1975	1267	1003
1976	1386	1110

or

Arranged by Variables		
(year)	1975	1976
(gdp)	1267	1386
(cons)	1003	1110

Figure R3.27 Data File Arranged by Observations or by Variables

You would read the one on the left using

```
READ      ; Nobs = 2 ; Nvar = 3 ; File = ... ; Names = … $
```

and the one on the right using

```
READ      ; By Variables ; Nobs = 2 ; Nvar = 3 ; File = ... ; Names = ... $
```

The data file would have three rows and two columns as on the right. The first row would be '1975 1976' and so on.

Regardless of the actual arrangement of the data file, Nobs is the number of observations, not necessarily the number of physical rows in the data set.

R3.5.6 Binary Files and Files from Other Programs

Most researchers use more than one program, and create data in a variety of environments. Every statistical program can read data files written by a few other programs – *LIMDEP* can read the several formats listed in this section – but none is able to read the data files written by every other program. This inability to pass data between some programs does place a constraint on some users, as there are several dozen statistical packages in use, and numerous data processing packages such as *Excel*. A partial solution is provided by packages that are designed specifically to convert data from a large variety of programs to any of them in turn.

Each of the major statistical packages in general use has its own 'native' system format. For *LIMDEP*, that is the .lpj project file described in Chapter R2; for Microsoft *Excel*, it is the .xls or .xlsx worksheet or workbook file; *SPSS* has its own .sav format, and so on. Software programs such as *StatTransfer* by Circle Systems, Inc. (http://www.stattransfer.com) can be used to convert system files from and to the native formats of many programs. This program can greatly facilitate your use of other packages with *LIMDEP*. With *StatTransfer*, you can convert native files from *SPSS*, *SAS*, *Stata*, *Excel*, *SYSTAT*, and about 25 other formats, to and from *LIMDEP* project files. The menu of file types supported by *StatTransfer* is shown in Figure R3.28.

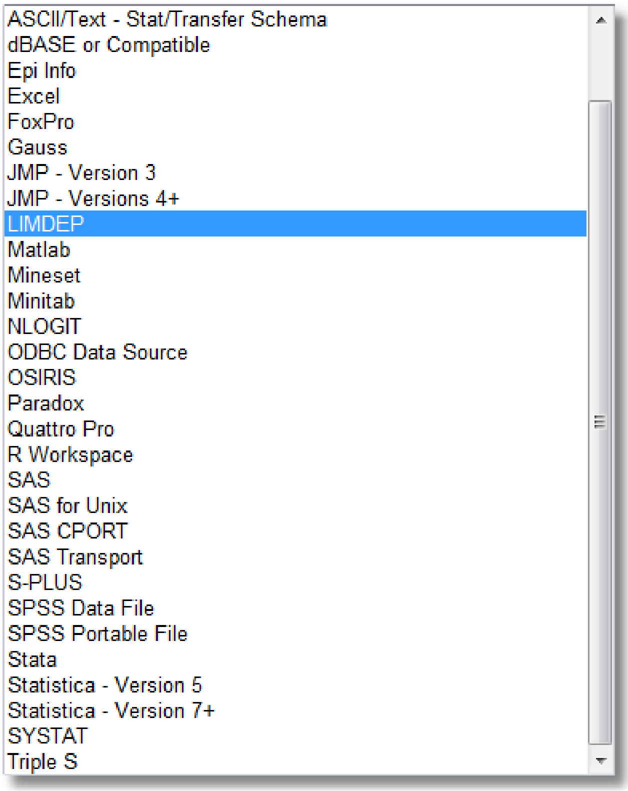


Figure R3.28 *StatTransfer* File Types

LIMDEP can read the following file formats:

Stata Files

LIMDEP can read the native file format (.dta) of *Stata* Versions 10 and 11. To read a *Stata* file, use

READ ; File = … ; Format = DTA $

If the file contains a column of descriptors or labels, add

; Labels = the column number

to the **READ** command, even if the observation labels are in the first column.

NOTE: The DTA format has changed over time. *LIMDEP* Version 11 supports the file format for *Stata* Versions 10 and 11. We cannot guarantee that this will continue to work with later versions of *Stata* (and we are aware that it does not work with some earlier versions).

Excel XLS Files

You can read a file written by *Excel* 2003 or earlier in the .xls format with

READ ; File = … ; Format = XLS $

Add **; Labels = the column number** if the file contains observation labels, even if the labels are included in the first column.

Note, once again, the .xls compatible files that *Excel* 2007 or later writes are not readable. You should generally not use the .xls format. If you have access to *Excel* (any version), open the file in *Excel* and use Save As to convert the file to the .csv format. (The only situation we see in which you will require this format is if you need to open the .xls file, but you do not have access to *Excel*.)

Binary Data Files

(If you are unsure if this is the right format for your data set, then almost certainly it is not.) If you are using a data set written in *binary* format, you must read it with

; Format = Binary

The other parts of the **READ** command are unchanged.

No other changes are necessary. But, note that the initial input of this file will be into an array of eight byte words. You must know in advance whether your file contains single or double precision (four or eight byte) values. The default file format is assumed to be four byte, single precision input, with conversion to double precision when the data are read. You can inform *LIMDEP* that your binary file is double precision with

; Double

You can extract a subset of the variables in a binary file. To put this in context, consider analyzing the data stored in a very large binary file, say 10,000 observations on 500 variables. You can extract selected variables from the master file. To extract a subset of variables from a binary file, use

```
READ        ; Nvar = nvar ; Names = ... ; File = name
            ; Format = Binary
            ; Size = width
            ; Cols = cols to read $
```

The specification of the particular columns to read may be an item by item list or ranges of columns, or a mix, as in

; Size = 500 ; Cols = 1, 4, 10-55, 60-100, 121, 129 $

R3.5.7 Formatted ASCII Files

You may be using data which are formatted in the file according to some uniform structure, particularly if the data set is large. Such files may require a detailed set of instructions for how to read them. For example, each of the two observations in the small file below could be a single 12 digit variable or 12 one digit variables, two six digit numbers, etc. Without further instructions, there is no way to tell.

```
197512671003
197613861110
```

A formatting code is used to lay out what data are contained in this file. Formatting allows you to save space by not having to include blanks or commas. You can read data according to a format by adding the specification

; Format = (format codes)

to the **READ** command. The format must be enclosed in parentheses. For example, the file above might be read with

```
READ        ; Names = ... ; Nvar = 3 ; Nobs = 2 ; Format = (F6.0,2F3.0) ; File = ... $
```

Details on the syntax of formatting codes are given below.

Format Codes

A format is used to describe how your data are arranged in each observation, character by character. For example, the data record

1234.56121213.4AC567

consists of 20 characters which can be grouped in many different ways to produce different sets of numbers. The format description is used to tell *LIMDEP* how to group the data in a set of values. Its general appearance is

(code, code, code, ...).

The format codes, or descriptors which you will normally use are

- Fw.d – the field is w characters wide, place d digits after the decimal point.
- X – ignore the character in this position.
- nX or nFw.d – n is a repetition factor. The X format must always be preceded by a repetition, even if it is 1.
- n(group of codes) – group of codes repeated n times.

To read the preceding string as the set of values '**1.23 4.56 1.21 2 13 .4 567**' and skip over the **AC**, which is not a number and therefore cannot be read by *LIMDEP*, you would use

(F3.2,F4.2,F3.2,F1.0,F2.0,F2.1,2X,F3.0)

Notice that the 1.23 is created by placing a decimal point between the 1 and the 2, while the 4.56 is read directly, and already contains a decimal point.

Repetition and grouping can save a lot of space. Note how the repetition of 2 is used to skip over the two letters. For another example, suppose you wanted to read the string 561212 as 56,1,21,2. This is a pair of two digit then one digit sequences. You could format it as

(...,2(F2.0,F1.0),...).

A useful result is that if the number in a field actually contains a decimal point, then the '**.d**' part of the format code is overridden. Thus, in the first example, while 13, .4 was read as 13 and .4 using F2.0,F2.1, it could have been read as 2F2.0. The presence of the decimal point in the second value would have overridden the specification of 0 digits after the decimal point in the format code. (But, using 2F2.1 would not be correct because though the second value would be correct, the first would be read as 1.3, not 13.)

Another useful descriptor is the slash format, '/.' You may need this if your data require more than one line per observation. This code means 'go to the next line and continue reading.' For example,

1234567812
3456

could be read as the numbers 123.456, 78.12, and 345.6 with the format **(F6.3,F4.2 / F4.1)**.

There are settings in which you do not need to provide the '/' format even if your observations take more than one line. Consider, for example, reading the preceding as a set of two digit numbers, 12, 34, 56, 78, 12, 34, and 56. The effect of the format **(5F2.0)** would be as follows: You are trying to read seven numbers, but you have only provided five format codes. The reader gets to the end of the five format codes and finds that it has two values yet to read. It drops to the next line of data and begins reading with the code at the beginning of the format statement.

WARNING: In specifying the format, use only *real* format codes, Fw.d. Never use integer formats – Iw – or character formats – Aw – for reading data. If data are coded with an exponential format, Ew.d, you can use the Fw.d code in *LIMDEP*. It handles exponential data as well.

Specifically Converting Blanks to Missing Values

Formatted **READ** commands always convert blanks to 0s. Since 0 is a legitimate value, if your blanks represent missing data (-999s), you need a method of requesting *LIMDEP* to make the conversion. This operation can be requested by using

; Blanks

in your *formatted* **READ** command. *You must provide the format statement.* If you use this option, your **READ** will be slower than otherwise, but the **READ** need never be done more than once. (Use **SAVE**.) This option also makes specifying the format codes a little easier. You can usually omit the '**.d**' in your specifications. For example:

To Read	As	Without option	With option
1234	1234	F4.0	F4
1.23	1.23	F4.2	F4
1234	12.34	F4.2	F4.2

If the decimal point is implicit, as in the third row, you must tell *LIMDEP* where to put it. The earlier example would be **(F4,2(1X,F4))**. With this option on, all blanks, nonnumeric data such as the word 'missing,' and fields containing only a period are converted to -999. Finally, if you require certain numeric values, such as -7, to be read as missing values, (i.e., converted to -999), simply **READ** them as they are, then use **RECODE** to do the conversion. (See Chapter R4.)

R3.5.8 Recoding Character Data

LIMDEP has a limited capability to manipulate character data. On input, you can recode a character symbol to a useable numeric value by using **; Recode** in your **READ** command. The setting involves a variable that is coded with a specific alphanumeric code which is converted to a numeric one. For example:

Northeast	1
South	2
Midwest	3
West	4

The recoding scheme is indicated on the **READ** command with

```
READ        ; Recode: variable(string = value, string = value, … ) /
                      variable(string = value, string = value, … )
```

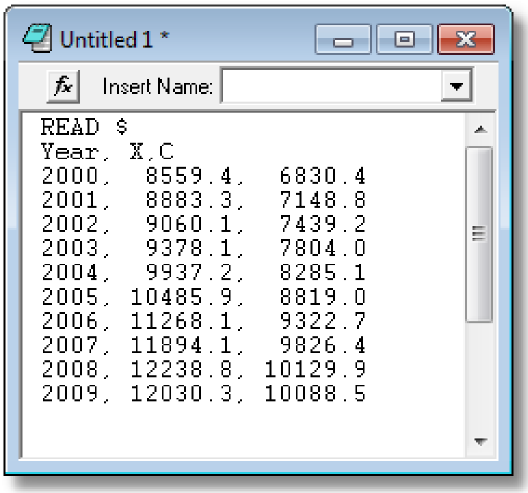
and so on. Note that the specification **; Recode** is followed by a colon; variable recodings are separated by a slash, and the transformations are separated by commas. You may recode as many variables as you like in this fashion. Character strings may be up to 20 characters. As always, upper case is the same as lower case, so you cannot use case to form different character strings. Any strings found in the data set that are not given in the list for the variable are converted to a missing value. For example, with a transformation of *sex* from 'female' to 1 and 'male' to 0, we might have

```
READ        ; Nobs   = …
            ; Nvar   = …
            ; File   = …
            ; Names = … , region,sex, …
            ; Recode: region(Northeast = 1, South = 2, Midwest = 3, West = 4) /
                     sex (female = 1, male = 0) $
```

This option may be used with ASCII text files such as .csv and .txt.

R3.6 Using the Text Editor as a Data File

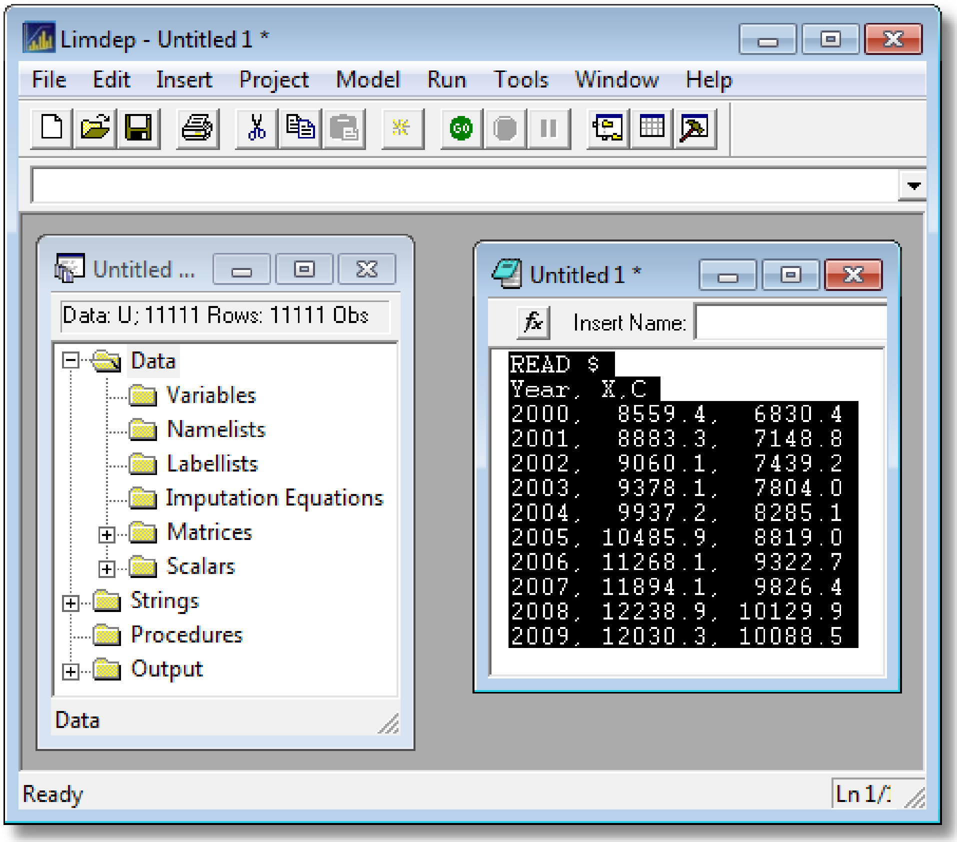
Your text/command editor (editing window) is actually a 'file' that *LIMDEP* reads when you click the GO button, so you can create data files in the text editor. Figure R3.29 shows a text editor in which we have entered a small data file with a **READ** command. (The data are Table F1.1 from Greene (2012).)



```
READ $
Year, X,C
2000,  8559.4,  6830.4
2001,  8883.3,  7148.8
2002,  9060.1,  7439.2
2003,  9378.1,  7804.0
2004,  9937.2,  8285.1
2005, 10485.9,  8819.0
2006, 11268.1,  9322.7
2007, 11894.1,  9826.4
2008, 12238.8, 10129.9
2009, 12030.3, 10088.5
```

Figure R3.29 Data in Text Editor

The data are in the text editor, but they are not in the program's data area yet. To read them, you would just highlight all the lines in the editor screen and click GO. (See Figure R3.30.) Note the base structure of the operation. The command is **READ $** which, without other information says that some data will follow, in the form of a row of names, followed by several rows of data. (In the data set above, the data are neatly lined up in columns. This is done here only for readability. It is not necessary.) Note, for present purposes, **READ** is the same as **IMPORT**, and you could use **IMPORT** as the command in the editor above.



```
READ $
Year, X,C
2000,  8559.4,  6830.4
2001,  8883.3,  7148.8
2002,  9060.1,  7439.2
2003,  9378.1,  7804.0
2004,  9937.2,  8285.1
2005, 10485.9,  8819.0
2006, 11268.1,  9322.7
2007, 11894.1,  9826.4
2008, 12238.9, 10129.9
2009, 12030.3, 10088.5
```

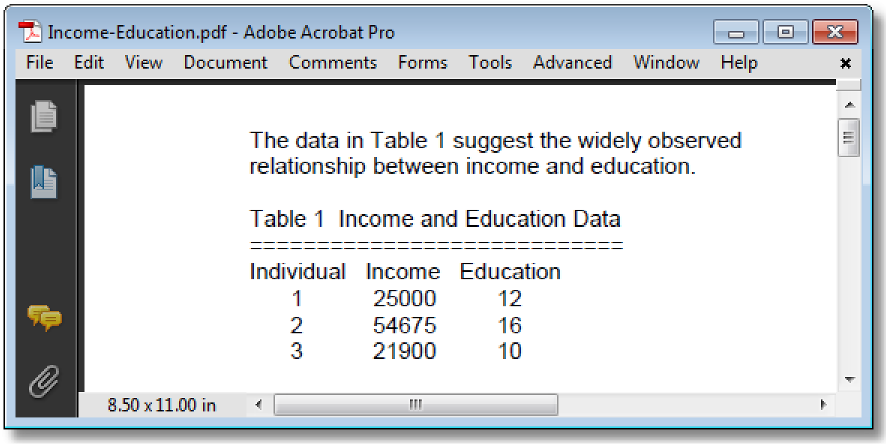
Figure R3.30 Reading Data from the Text Editor

R3.6.1 Use the Text Editor to Avoid Creating a Data File

In addition to reading data off the screen in the editor, you can **y**ou can pull data directly out of documents, such as *Word* or PDF files, and import them into *LIMDEP* without having to put them in a data file first. To import the data into *LIMDEP*, do the following:

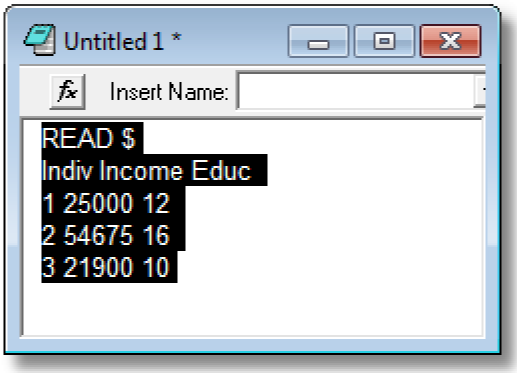
1. Open a text editing window and place the command **READ $** in the first line.
2. Highlight the data you wish to copy including column headers and use Edit:Copy to place them on the clipboard.
3. Return to the text editor in *LIMDEP* and use Edit:Paste to paste the data under the **READ** command.
4. Highlight the **READ** command, the column header names and the data (or use Edit:Select All) and click GO.

Figures R3.31 and R3.32 below show this operation with a PDF file. You may import any data set this way, though for more than a few hundred observations, it may be a bit cumbersome.


Income-Education.pdf - Adobe Acrobat Pro
File Edit View Document Comments Forms Tools Advanced Window Help
8.50 x 11.00 in

```
The data in Table 1 suggest the widely observed
relationship between income and education.

Table 1  Income and Education Data
============================
Individual   Income   Education
    1         25000      12
    2         54675      16
    3         21900      10
```

Figure R3.31 Data in a PDF File

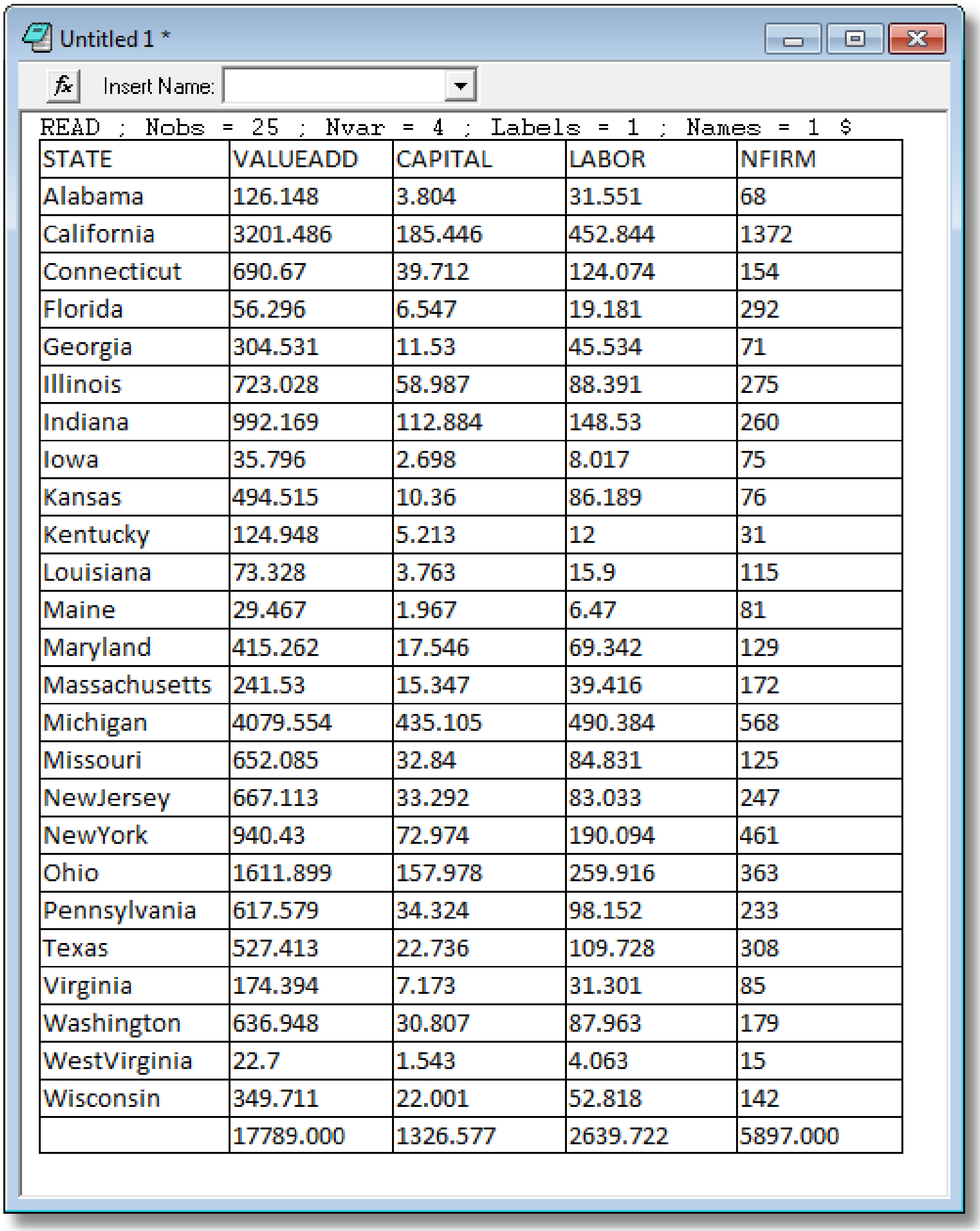


```
READ $
Indiv Income Educ
1 25000 12
2 54675 16
3 21900 10
```

Figure R3.32 Data Transported to the Text Editor

R3.6.2 Exporting from *Excel* to the Text Editor

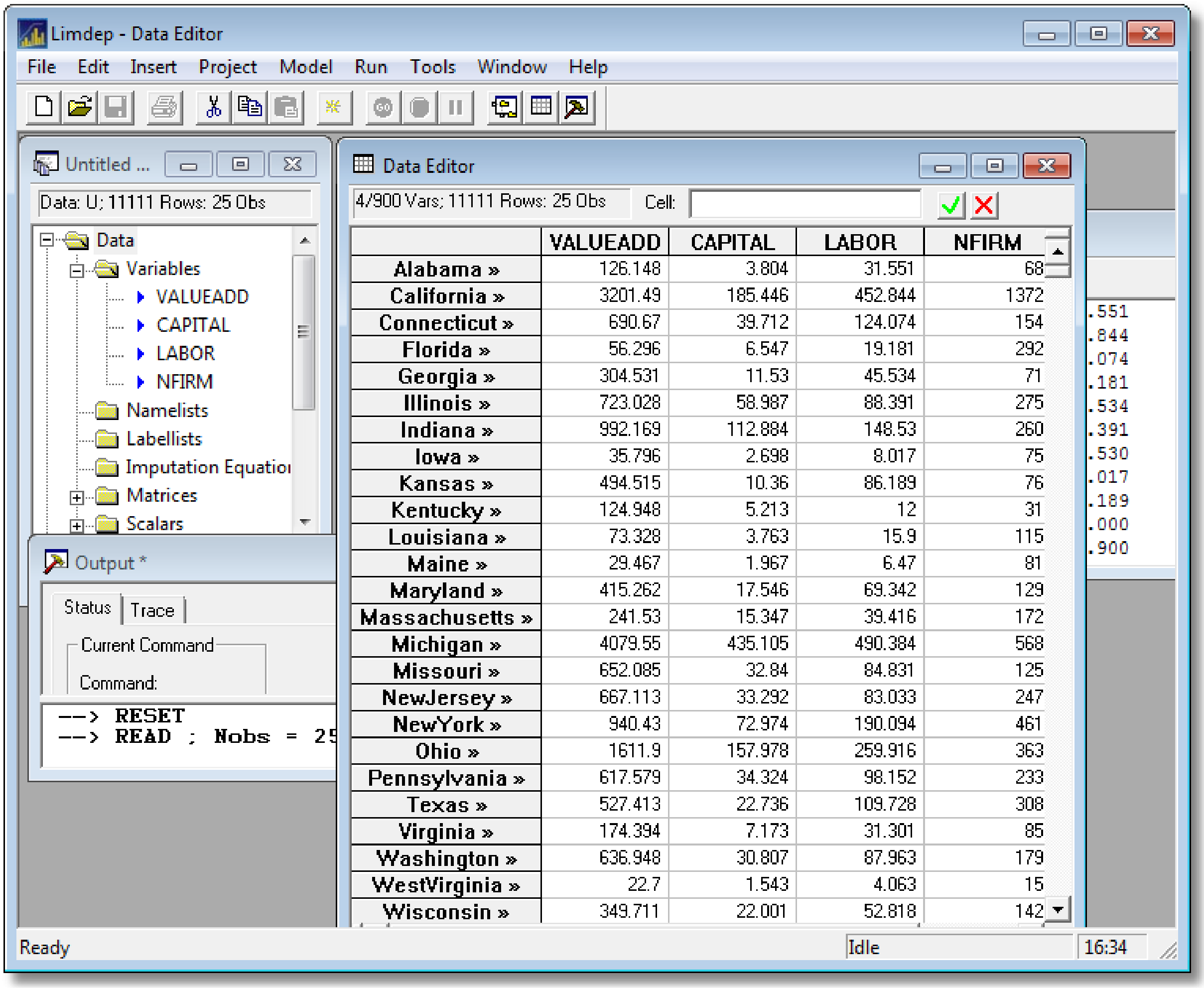
As we examined above, you can copy/paste data directly from other programs into *LIMDEP*'s text editor. This is a quick way to read a small data set. You can also copy/paste data from a spreadsheet program into the text editor. Figure R3.33 shows an example. We have copied the cells from *Excel* that contain the full transportation data set shown in Figure R3.10, including labels and variable names, and pasted them into the *LIMDEP* text editor under the **READ** command. Note that the **READ** command does not specify a file name. To import these data just highlight all the lines including the **READ** command (or select Edit/Select All) and click GO. Figure R3.34 shows the result. (Note that the editor inherits the cell boundary lines. These are ignored as the data are read. They will also disappear if the text editor contents are written to a .lim file and later read back into the editor.)



```
READ ; Nobs = 25 ; Nvar = 4 ; Labels = 1 ; Names = 1 $
```

STATE	VALUEADD	CAPITAL	LABOR	NFIRM
Alabama	126.148	3.804	31.551	68
California	3201.486	185.446	452.844	1372
Connecticut	690.67	39.712	124.074	154
Florida	56.296	6.547	19.181	292
Georgia	304.531	11.53	45.534	71
Illinois	723.028	58.987	88.391	275
Indiana	992.169	112.884	148.53	260
Iowa	35.796	2.698	8.017	75
Kansas	494.515	10.36	86.189	76
Kentucky	124.948	5.213	12	31
Louisiana	73.328	3.763	15.9	115
Maine	29.467	1.967	6.47	81
Maryland	415.262	17.546	69.342	129
Massachusetts	241.53	15.347	39.416	172
Michigan	4079.554	435.105	490.384	568
Missouri	652.085	32.84	84.831	125
NewJersey	667.113	33.292	83.033	247
NewYork	940.43	72.974	190.094	461
Ohio	1611.899	157.978	259.916	363
Pennsylvania	617.579	34.324	98.152	233
Texas	527.413	22.736	109.728	308
Virginia	174.394	7.173	31.301	85
Washington	636.948	30.807	87.963	179
WestVirginia	22.7	1.543	4.063	15
Wisconsin	349.711	22.001	52.818	142
	17789.000	1326.577	2639.722	5897.000

Figure R3.33 *Excel* Spreadsheet Data Copied to the Text Editor



```
--> RESET
--> READ ; Nobs = 25
```

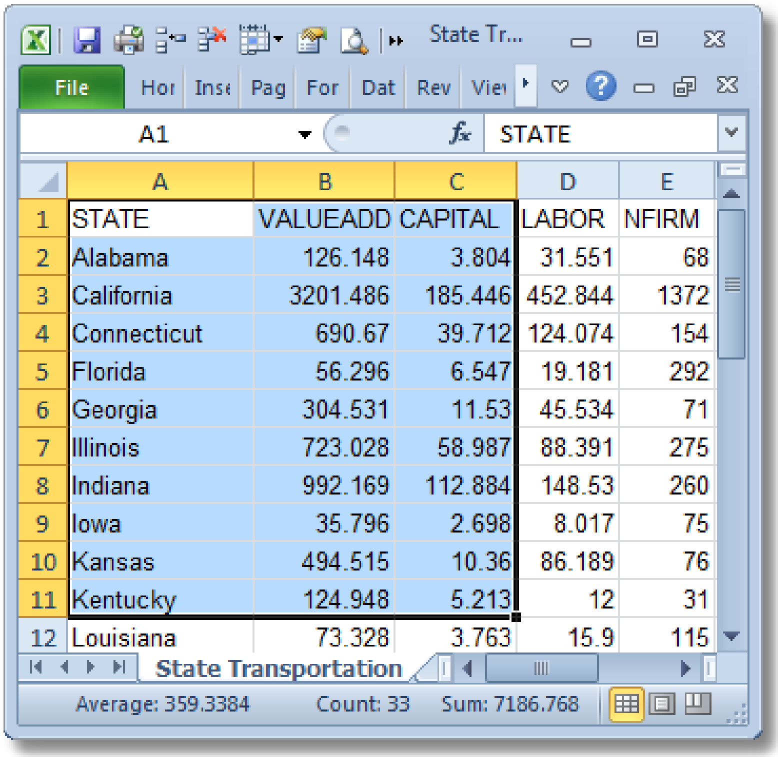
	VALUEADD	CAPITAL	LABOR	NFIRM
Alabama »	126.148	3.804	31.551	68
California »	3201.49	185.446	452.844	1372
Connecticut »	690.67	39.712	124.074	154
Florida »	56.296	6.547	19.181	292
Georgia »	304.531	11.53	45.534	71
Illinois »	723.028	58.987	88.391	275
Indiana »	992.169	112.884	148.53	260
Iowa »	35.796	2.698	8.017	75
Kansas »	494.515	10.36	86.189	76
Kentucky »	124.948	5.213	12	31
Louisiana »	73.328	3.763	15.9	115
Maine »	29.467	1.967	6.47	81
Maryland »	415.262	17.546	69.342	129
Massachusetts »	241.53	15.347	39.416	172
Michigan »	4079.55	435.105	490.384	568
Missouri »	652.085	32.84	84.831	125
NewJersey »	667.113	33.292	83.033	247
NewYork »	940.43	72.974	190.094	461
Ohio »	1611.9	157.978	259.916	363
Pennsylvania »	617.579	34.324	98.152	233
Texas »	527.413	22.736	109.728	308
Virginia »	174.394	7.173	31.301	85
Washington »	636.948	30.807	87.963	179
WestVirginia »	22.7	1.543	4.063	15
Wisconsin »	349.711	22.001	52.818	142

```
.551
.844
.074
.181
.534
.391
.530
.017
.189
.000
.900
```

Figure R3.34 *Excel* File Read into *LIMDEP* from the Text Editor

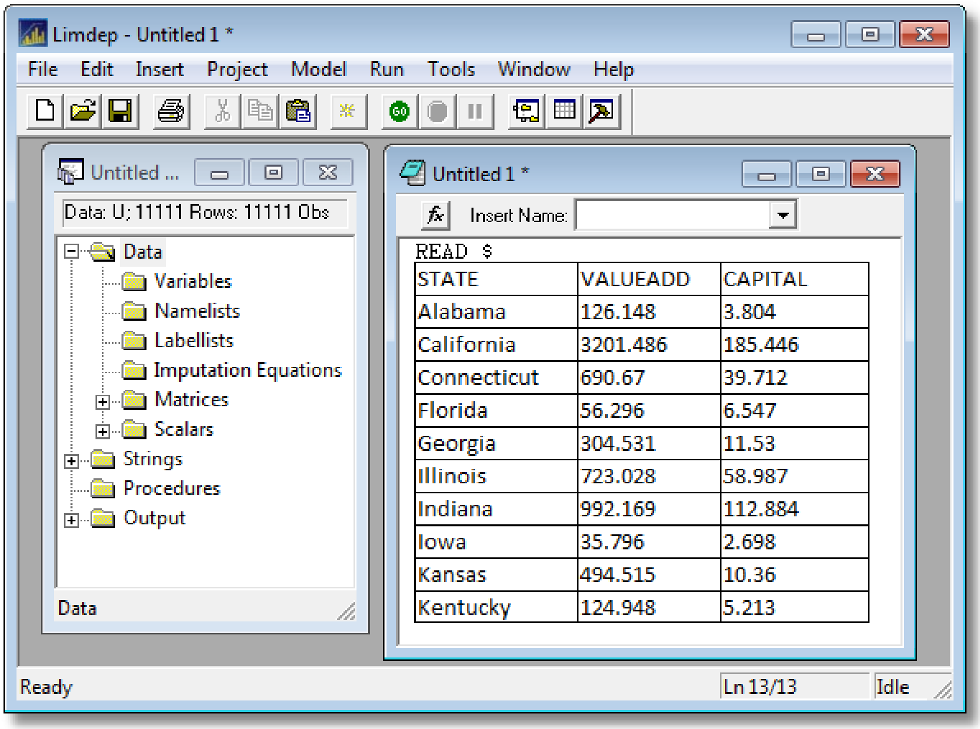
You can also copy/paste a portion of a spreadsheet data set into the text editor. Figure R3.35 shows an example. We have selected part of the transportation data. The highlighted range is just copied in *Excel* and pasted into the *LIMDEP* text editor, with the result shown in Figure R3.36. The data are then read into *LIMDEP* just by highlighting the **READ** command and the table, and clicking GO.

We note one possible advantage of this procedure. When you use Save As (or Save), *Excel* saves the entire spreadsheet, not a selected piece of it. This device provides a shortcut to exporting a part of a spreadsheet. The alternative within *Excel* is to open another spreadsheet, paste the cells into that new spreadsheet and save it as a separate file. Then you would import that smaller spreadsheet. This saves you a step.



	A	B	C	D	E
1	STATE	VALUEADD	CAPITAL	LABOR	NFIRM
2	Alabama	126.148	3.804	31.551	68
3	California	3201.486	185.446	452.844	1372
4	Connecticut	690.67	39.712	124.074	154
5	Florida	56.296	6.547	19.181	292
6	Georgia	304.531	11.53	45.534	71
7	Illinois	723.028	58.987	88.391	275
8	Indiana	992.169	112.884	148.53	260
9	Iowa	35.796	2.698	8.017	75
10	Kansas	494.515	10.36	86.189	76
11	Kentucky	124.948	5.213	12	31
12	Louisiana	73.328	3.763	15.9	115

Figure R3.35 Data in *Excel*



```
READ $
```

STATE	VALUEADD	CAPITAL
Alabama	126.148	3.804
California	3201.486	185.446
Connecticut	690.67	39.712
Florida	56.296	6.547
Georgia	304.531	11.53
Illinois	723.028	58.987
Indiana	992.169	112.884
Iowa	35.796	2.698
Kansas	494.515	10.36
Kentucky	124.948	5.213

Figure R3.36 *Excel* Cells Copied to Text Editor

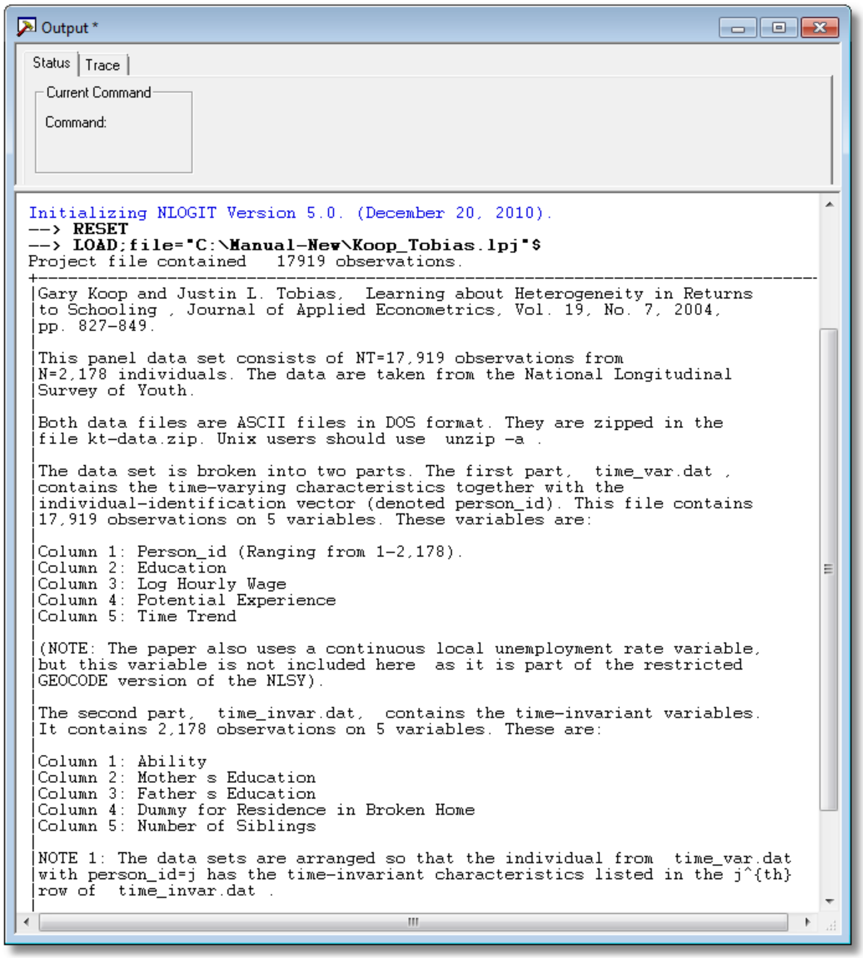
R3.7 Documenting the Contents of a Data/Project File

After you have imported your data, it is a good idea to save it immediately as a project file. (See Section R2.2.) You need only import the data set once. Thereafter, when necessary, you should reload the data just by opening the project file. This is a much faster way to import the data into a new session. Saving the project is *LIMDEP*'s 'Save' operation.

It may be useful to carry documentation of the data set in a project permanently with the file. For example, Figure R3.37 shows some text information about the Koop and Tobias data set used in our example in Section R5.4. The text description is saved permanently in the project file. You can set this up as follows: In any text editing window, place the following text information

DATA
… Up to 255 lines containing up to 80 characters on each line. …
ENDATA

Use Edit/Select All or just highlight the entire script and click GO to execute it. Later, when you save the project file, this information is saved with it, permanently. When you reload the project, your codebook information will be displayed in the output screen.



```
Initializing NLOGIT Version 5.0. (December 20, 2010).
--> RESET
--> LOAD;file="C:\Manual-New\Koop_Tobias.lpj"$
Project file contained   17919 observations.
+---------------------------------------------------------------------------
|Gary Koop and Justin L. Tobias,  Learning about Heterogeneity in Returns
|to Schooling , Journal of Applied Econometrics, Vol. 19, No. 7, 2004,
|pp. 827-849.
|
|This panel data set consists of NT=17,919 observations from
|N=2,178 individuals. The data are taken from the National Longitudinal
|Survey of Youth.
|
|Both data files are ASCII files in DOS format. They are zipped in the
|file kt-data.zip. Unix users should use  unzip -a .
|
|The data set is broken into two parts. The first part,  time_var.dat ,
|contains the time-varying characteristics together with the
|individual-identification vector (denoted person_id). This file contains
|17,919 observations on 5 variables. These variables are:
|
|Column 1: Person_id (Ranging from 1-2,178).
|Column 2: Education
|Column 3: Log Hourly Wage
|Column 4: Potential Experience
|Column 5: Time Trend
|
|(NOTE: The paper also uses a continuous local unemployment rate variable,
|but this variable is not included here  as it is part of the restricted
|GEOCODE version of the NLSY).
|
|The second part,  time_invar.dat,  contains the time-invariant variables.
|It contains 2,178 observations on 5 variables. These are:
|
|Column 1: Ability
|Column 2: Mother s Education
|Column 3: Father s Education
|Column 4: Dummy for Residence in Broken Home
|Column 5: Number of Siblings
|
|NOTE 1: The data sets are arranged so that the individual from  time_var.dat
|with person_id=j has the time-invariant characteristics listed in the j^{th}
|row of  time_invar.dat .
|
```

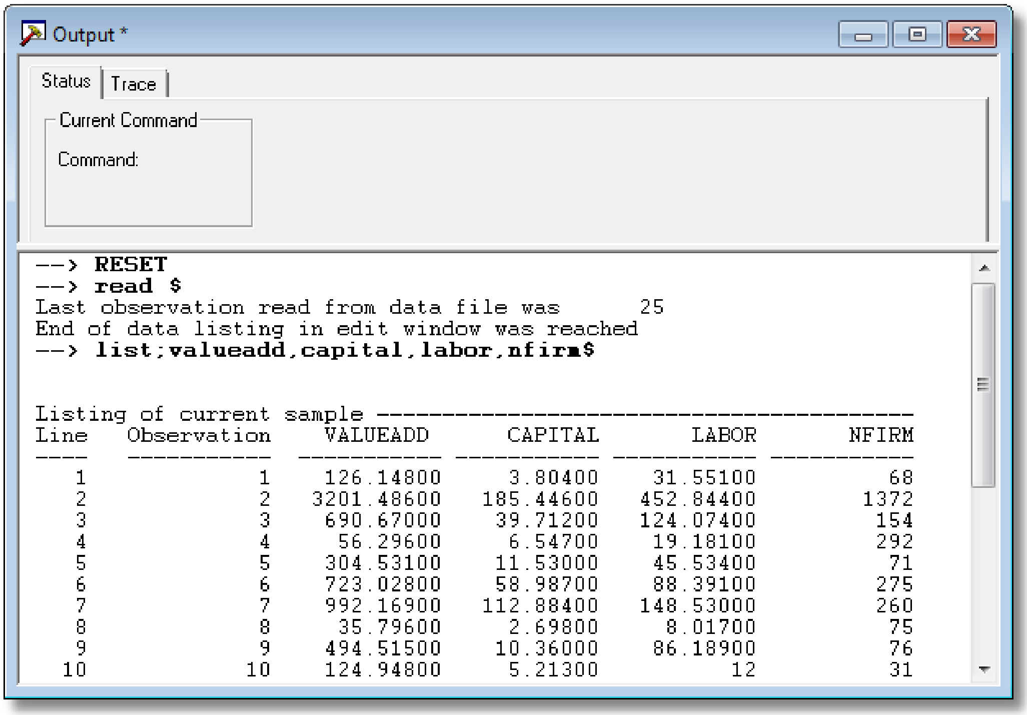
Figure R3.37 Data Documentation Saved in Project File

R3.8 Listing Data in Your Output Window

The **LIST** command is used to send a listing of the current sample of observations on a particular set of variables to the screen. The command is

LIST ; ... list of variables $

Figure R3.38 shows an example.

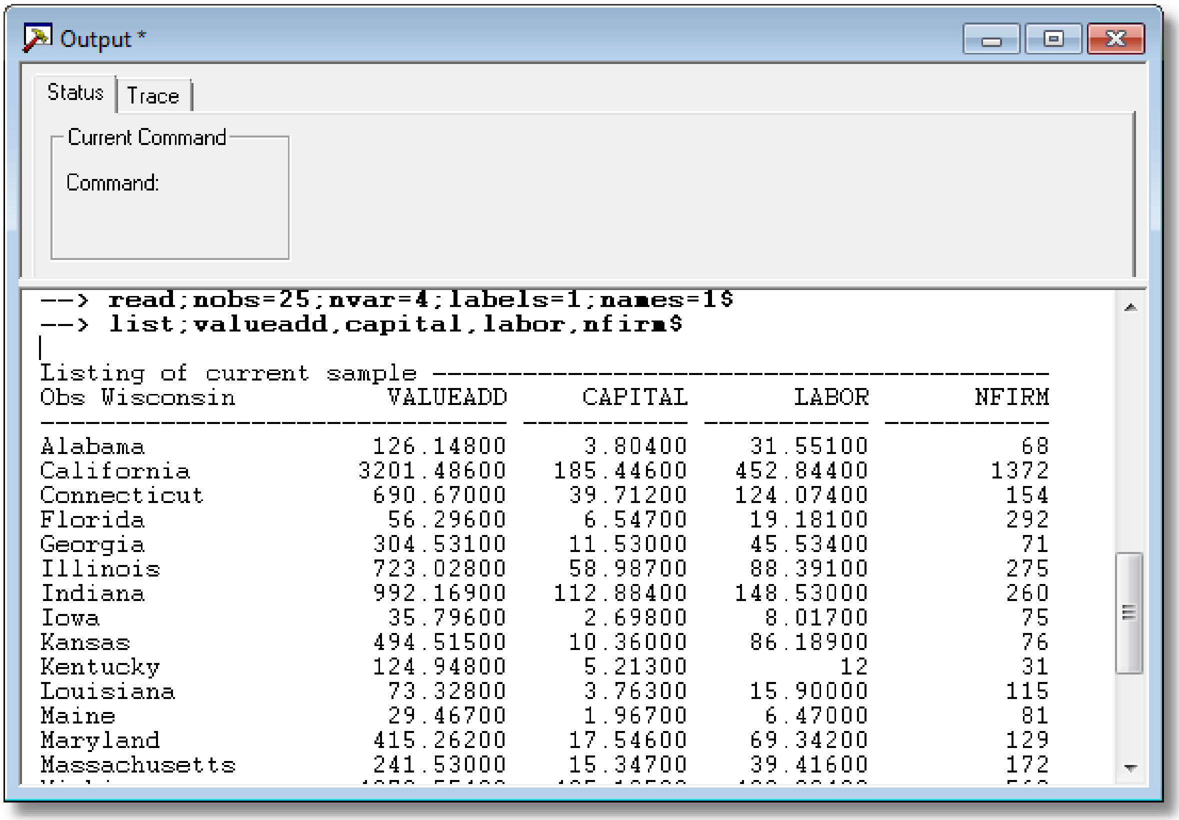


```
--> RESET
--> read $
Last observation read from data file was      25
End of data listing in edit window was reached
--> list;valueadd,capital,labor,nfirm$


Listing of current sample ----------------------------------------
Line    Observation    VALUEADD     CAPITAL       LABOR       NFIRM
----    -----------  -----------  -----------  -----------  -----------
   1              1    126.14800      3.80400     31.55100           68
   2              2   3201.48600    185.44600    452.84400         1372
   3              3    690.67000     39.71200    124.07400          154
   4              4     56.29600      6.54700     19.18100          292
   5              5    304.53100     11.53000     45.53400           71
   6              6    723.02800     58.98700     88.39100          275
   7              7    992.16900    112.88400    148.53000          260
   8              8     35.79600      2.69800      8.01700           75
   9              9    494.51500     10.36000     86.18900           76
  10             10    124.94800      5.21300           12           31
```

Figure R3.38 Output from LIST

The listing of the data includes the observation number in the current sample and the row number in the actual data set. These will be the same unless you have selected a subsample to list. For example, if you selected for observation only those states with value added greater than 500, the listing of states in Figure R3.38 would have had lines numbered 1,2,3,4 but observation numbers 2,3,6,7.

If your data were read with observation labels, then the labels will replace the line numbers shown. For the data above, for example, we would have the display in Figure R3.39.



```
--> read;nobs=25;nvar=4;labels=1;names=1$
--> list;valueadd,capital,labor,nfirm$

Listing of current sample ---------------------------------------
Obs Wisconsin              VALUEADD     CAPITAL       LABOR       NFIRM
------------------------------ ----------- ----------- -----------
Alabama                  126.14800     3.80400    31.55100          68
California              3201.48600   185.44600   452.84400        1372
Connecticut              690.67000    39.71200   124.07400         154
Florida                   56.29600     6.54700    19.18100         292
Georgia                  304.53100    11.53000    45.53400          71
Illinois                 723.02800    58.98700    88.39100         275
Indiana                  992.16900   112.88400   148.53000         260
Iowa                      35.79600     2.69800     8.01700          75
Kansas                   494.51500    10.36000    86.18900          76
Kentucky                 124.94800     5.21300          12          31
Louisiana                 73.32800     3.76300    15.90000         115
Maine                     29.46700     1.96700     6.47000          81
Maryland                 415.26200    17.54600    69.34200         129
Massachusetts            241.53000    15.34700    39.41600         172
```

Figure R3.39 Data Listing with Observation Labels

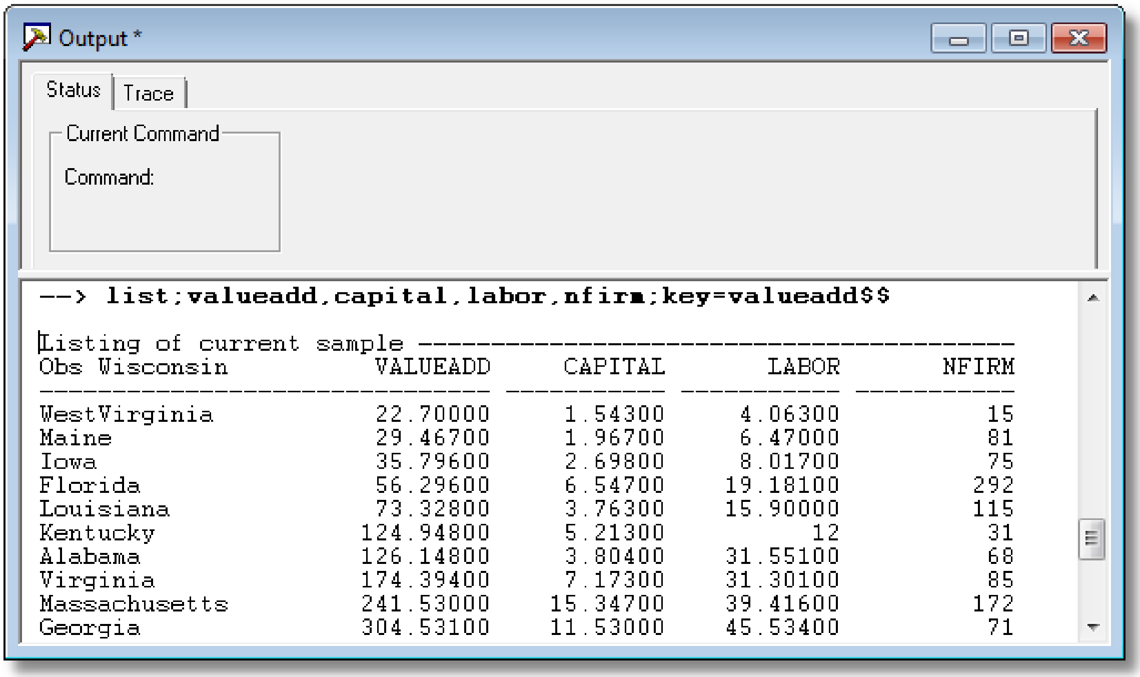
You can request that your listed data be sorted by a particular variable by using

LIST ; ... list of variables ; Key = a variable $

The 'Key' variable is sorted carrying the listed variables with it, then the listing is produced with the sorted data. Figure R3.40 shows an example based on Figure R3.39. Two important notes:

- The sorting does not affect the data in your data area. The data to be listed are copied, then sorted, then the sorted copies are displayed. The original data are untouched. The **SORT** command may be used if you wish to sort your data.

- The key variable does not have to be one of the variables being listed.



```
--> list;valueadd,capital,labor,nfirm;key=valueadd$$

Listing of current sample ----------------------------------------
Obs Wisconsin                 VALUEADD     CAPITAL       LABOR       NFIRM
------------------------------ ----------- ----------- ----------- -----------
WestVirginia                  22.70000     1.54300     4.06300          15
Maine                         29.46700     1.96700     6.47000          81
Iowa                          35.79600     2.69800     8.01700          75
Florida                       56.29600     6.54700    19.18100         292
Louisiana                     73.32800     3.76300    15.90000         115
Kentucky                     124.94800     5.21300          12          31
Alabama                      126.14800     3.80400    31.55100          68
Virginia                     174.39400     7.17300    31.30100          85
Massachusetts                241.53000    15.34700    39.41600         172
Georgia                      304.53100    11.53000    45.53400          71
```

Figure R3.40 Data Listing Sorted by Key Variable

R3.9 Exporting and Writing Data Files

The essential commands to create a new data file are **EXPORT** and **WRITE**. **EXPORT** is used to create a CSV file that can be imported directly into *Excel* or other programs that recognize this format. **WRITE** is used to produce a simple text file with numbers stacked in columns, possibly using a format that you specify.

R3.9.1 How to EXPORT a CSV File

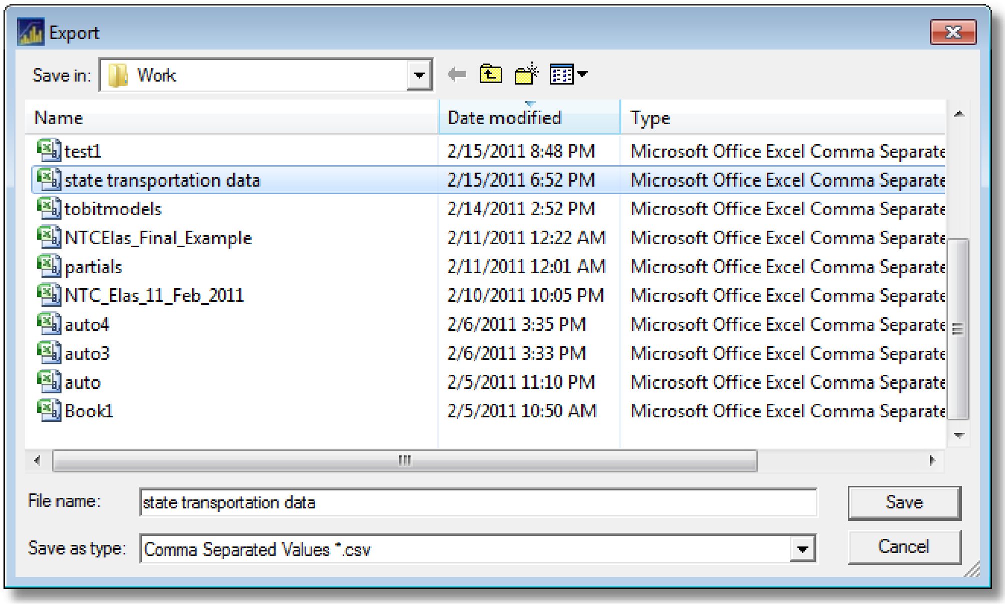
The command for exporting a file is

EXPORT ; list of variables ; File = filename $

No other information is given with this command. The observations written in the file are those defined by the current sample. (The current sample is discussed in Chapter R7.) The file will have the CSV format and will be named filename.csv. If you have put an extension on the filename, it will be replaced. Other formatting conventions are

- Variable names and observation labels will be exported as well as the data.
- Missing values will appear appropriately as blanks in the file.
- Integers will be exported as integers, though they are stored as reals inside *LIMDEP*.
- Noninteger values are written with seven significant digits.

To export a data file using the menu option, first select Project:Export/Variables in the project menu to select the file, as shown in Figure R3.41. (You can also select Export Variables from the right mouse menu in the data editor to open the same dialog box). The next step is to select the variables to be written in the file, either by selecting Select All or by selecting the variables by name in the window. (Select None is used if you wish to undo your selections and start over.) Figure R3.42 shows an example.



Name	Date modified	Type
test1	2/15/2011 8:48 PM	Microsoft Office Excel Comma Separate
state transportation data	2/15/2011 6:52 PM	Microsoft Office Excel Comma Separate
tobitmodels	2/14/2011 2:52 PM	Microsoft Office Excel Comma Separate
NTCElas_Final_Example	2/11/2011 12:22 AM	Microsoft Office Excel Comma Separate
partials	2/11/2011 12:01 AM	Microsoft Office Excel Comma Separate
NTC_Elas_11_Feb_2011	2/10/2011 10:05 PM	Microsoft Office Excel Comma Separate
auto4	2/6/2011 3:35 PM	Microsoft Office Excel Comma Separate
auto3	2/6/2011 3:33 PM	Microsoft Office Excel Comma Separate
auto	2/5/2011 11:10 PM	Microsoft Office Excel Comma Separate
Book1	2/5/2011 10:50 AM	Microsoft Office Excel Comma Separate

File name: state transportation data

Save as type: Comma Separated Values *.csv

Figure R3.41 Project:Export Menu

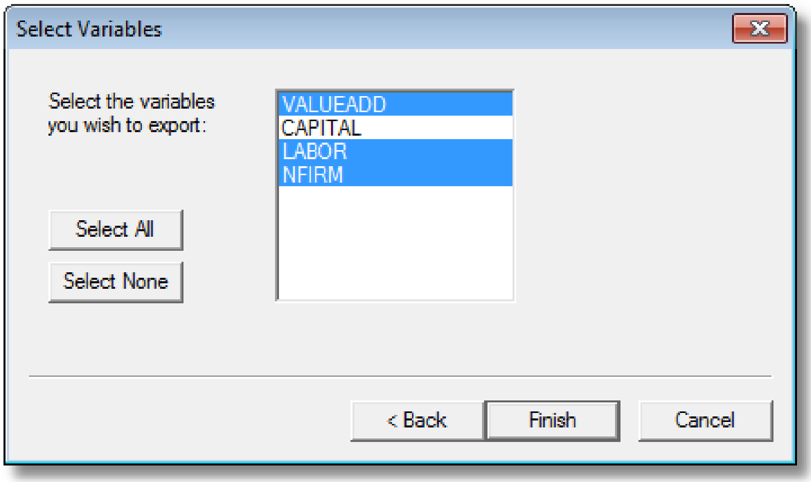


Figure R3.42 Project:Export Variables Specification

TIP: You can transfer data directly between *LIMDEP* and your spreadsheet programs, such as *Excel*. In the data editor, you can select a block of values by highlighting them, then use Edit:Copy in *LIMDEP* and Edit:Paste in *Excel* to replicate the block in *Excel*. This does not move the names, so you should begin your transfer in *Excel* in the second row, then enter the names in the first.

If you are creating more than one data file, it might be useful to put a running index in your export file. For example, if you are simulating multiple data sets and you want to have a separate CSV file for each, it will be useful to be able to distinguish the files. The following example shows how you could do this:

```
PROCEDURE $
CREATE       ; x = Rnn(0,1) $
EXPORT       ; x ; File = x#i#data.csv $
ENDPROCEDURE $
EXECUTE      ; i = 1,4 $
```

The procedure creates a column of random draws in each of four replications, and creates files x0001data.csv, x0002data.csv, x0003data.csv and x0004data.csv. The use of #number# expands the number into a four digit index (it may be anywhere in the filename). The index must be an integer from 0 to 9999. Note that without the surrounding #s, this procedure would just create xidata.csv, four times. You may expand an index number in the export file name in any context, not necessarily in a procedure.

R3.9.2 How to WRITE a Data File

```
WRITE        ; list of variables
             ; File = filename ; Format = the type desired $
```

NOTE: The observations written in the file are those defined by the current sample. The current sample is discussed in Chapter R7.

The data file written will contain only the numeric values from the data area. If you would like to include the variable names in the top row of the file, add

```
             ; Names
```

to the **WRITE** command. This command will write the variables listed in the file named using (6G14.6) format. The G14.6 format code provides a 14 column field, and six significant digits in the number. If the number written is too large or too small to write in this fashion, this format reverts to a scientific notation format, ±0.nnnnnnE±ee.

NOTE: Missing values are given the numeric value -999.

If you would prefer some other format for the file, you can specify one with

```
             ; Format = (your own format)
```

If, for example, you are writing binary variables, allowing 14 columns for a one digit number is a bit wasteful. *LIMDEP* cannot check the syntax of this format for you, so you may induce an error if you provide an improper format. Do remember to include the parentheses. The **WRITE** command will fail if your format contains an error.

You can also extend the formatting of a **WRITE** command to integers and at the same time, convert the missing values to the '.' convention used by other programs such as *SAS*. The alternative format specification

; Format = [… format …]

with the specification enclosed in square brackets instead of parentheses allows you these specifications:

Iw	= an integer of width w,
Fw.d	= real value, width w, d digits after the decimal point,
Ew.d	= exponential format (scientific notation),
nX	= skip n spaces.

This format may not have any embedded parentheses. When you use this option, missing values are automatically converted to dots in the file.

R3.10 Adding Observations – The APPEND Command

The **IMPORT** and **READ** commands are used to add variables, i.e., columns to your data set. If you wish to add observations, i.e., rows, you use the **APPEND** command, instead. The command structure is the same as **READ**, i.e., (optional parts are in brackets)

```
APPEND        ; File = name of file
               ; Nvar = number of variables
               ; Nobs = number of observations
              [; Format = (the format)or CSV or XLS or Binary]
              [; Names = number or list of names]
              [; By Variables]  $
```

The command works as follows: *LIMDEP* keeps a pointer which indicates where the next data file to be read should be placed. Thus, before you **READ** any data, the pointer equals 1. If you initially **READ** your first data set of, say, 25 observations, the pointer is reset to 26. Each time you **APPEND** a data set, the pointer is advanced. It is also advanced if you **READ** a longer data set after a shorter one. The pointer always points to the row after the bottom of your data. Your first **READ** is equivalent to an **APPEND**. Thereafter, if your command is **APPEND** instead of **READ**, the data are read as usual, but placed in the data area in the rows beginning at the pointer, instead of at the top.

Columns are handled as follows: Suppose you **READ** 25 observations on *x*, *y*, and, *z*. Now, the command

```
APPEND        ; File = ... ; ...  ; Names = x,y,w ; Nobs = 15 $
```

will add 15 observations to *x* and *y*. Since *w* doesn't exist yet, a new variable is created for it. Since we are using the **APPEND** command, not **READ**, the 15 observations on this new variable are placed in rows 26-40, not 1-25. Rows 1-25 of *w* and rows 26-40 of *z* will contain missing values after this command is carried out.

Given the preceding, there are two ways an **APPEND** command can go wrong. If the data are stored internally, you may run out of rows. You can run out of columns either way. *LIMDEP* will take as much data as it can fit when an **APPEND** command is given. If the full data set doesn't fit, you will be warned. Finally, because of the way it is handled internally, you may only **APPEND** a total of 200,000 numbers at a time.

R4: Data Transformations

R4.1 Data Transformations

You will usually need to transform your data, for example to obtain logarithms, differences, or any number of other possibilities. *LIMDEP* provides all of the algebraic transformations you are likely to need with the **CREATE** command. It is often useful to recode a continuous variable into discrete values or to combine discrete values into a smaller number of groups, for example to prepare data for contingency tables. The **RECODE** command is provided for this purpose. You can use **SORT** to arrange one or more variables in ascending or descending order. The five commands described in this chapter are as follows:

CREATE	**; variable name = expression $** to create a transformed variable
DELETE	**; list of variables $** to delete variables from the data set
RECODE	**; variable ; range of values = new value ... $** to recode a variable
RENAME	**; old name = new name $** to change the name of a variable
SORT	**; Lhs = key variable [; Rhs = variables to carry] $**

CREATE also provides functions for rearranging data to create partitioned data matrices and random number generators for generating random samples.

R4.2 The CREATE Command

The **CREATE** command is used to modify existing variables or compute new ones. The essential syntax of the command is

```
CREATE        ; name = expression  $
```

Commands may be grouped in a single instruction, with

```
CREATE        ; name = expression  ;  name = expression; ... $
```

If you have a very large data set, the second form is preferable because each **CREATE** command requires a pass through the sample observations regardless of the number of subcommands. Unless you have hundreds of thousands of observations, however, the difference in computation speed will probably not be discernible. If your instruction is part of a long involved script, it might be useful to document the calculations as you do them. You can add a title to the command as follows:

```
CREATE        ; Title = the title to use ; name = expression ; … $
```

For example,

```
CREATE        ; Title = Random Numbers
              ; x = Rnn(0,1)
              ; y = x + Rnn(0,1) $
```

Transformations may also be made conditionally, as in

CREATE ; If (...) ... expressions $

The various forms of the conditional transformations are described in Section R4.2.2.

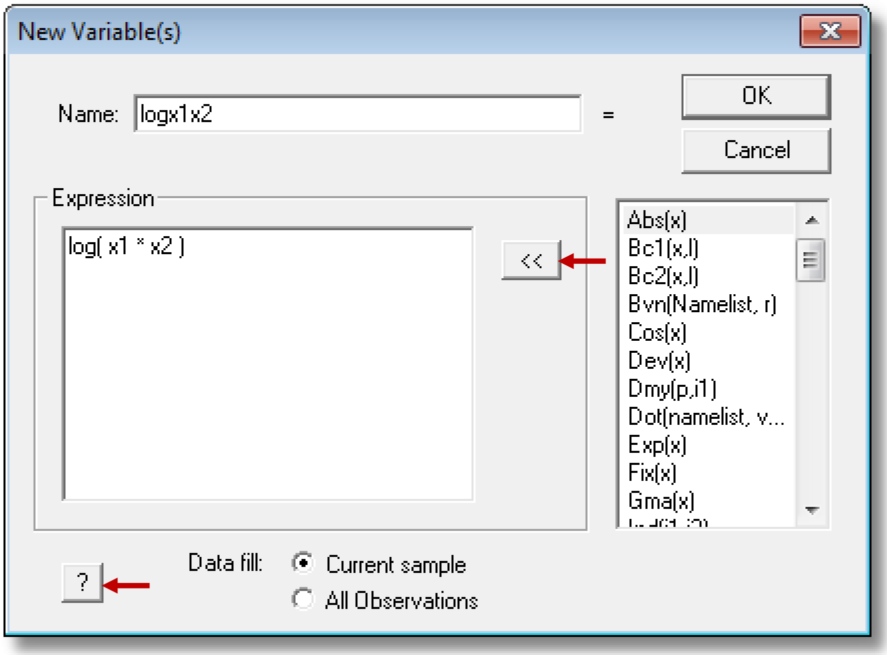
You may also enter your command in a dialog box, as shown in Figure R4.1. The dialog box is invoked by selecting Project:New/Variable or by going to the data editor and clicking the right mouse button which will bring up a menu that includes New Variable. You may now enter the name for the new or transformed variable in the Name window. If you click OK at this point without entering an expression for the variable, the new variable is created with all observations treated as missing. The equivalent command to the one in the dialog box would be

CREATE ; logx1x2 $ or **CLEAR ; logx1x2 $**

You can create more than one empty variable this way as well by giving a list of names separated by commas either in the dialog box or in a command. For example,

CREATE or CLEAR ; logx1x2, logx1x3, logx2x3 $

You may enter an expression for the new or transformed variable in the Expression window. Two other features to note in the dialog box are the query (?) button at the lower left, which will invoke the online Help file for **CREATE**, and the function insertion button at the right of the Expression window. You can select a function from the list in the window at the right of the dialog box, then insert that function in the Expression window by clicking the function insertion button. This allows some convenience in copying the function name into the small editing window, and also shows a listing of the function names you can use.



Name: logx1x2 =

Expression: log(x1 * x2)

Data fill: [✓ Current sample] All Observations

Figure R4.1 New Variable Dialog Box

A **CREATE** command operates on the 'current sample.' (See Chapter R7.) If this is a subset of the data, remaining observations will not be changed. If you are creating a new variable for the subset of observations, remaining observations will be undefined (missing). You can override this feature by using

```
CREATE ; Fill ; ... the rest of the command $
```

in your command. With this additional setting, the transformations listed will be applied to all observations in the data set, whether in the current sample or not. This is the Data fill option that appears at the bottom center of the dialog box in Figure R4.1.

You might want to apply a transformation to a specific subset of the data. Since **CREATE** operates on the current sample, one way to do so is to set the sample accordingly before the **CREATE** command (then reset it after). A more direct approach is to instruct **CREATE** directly to operate on the specific subset. For example, suppose the data set contains full time (fulltime = 1) and part time (fulltime = 0) workers, and the variable to be created is household income, but only for fulltime = 1. The command might appear

```
CREATE ; If [fulltime = 1] ; income = hhincome $
```

Observations for which fulltime = 0 are not affected by this command. The setting of the current sample is unchanged. Generally, any logical condition can appear in the square brackets. The example above will be typical.

R4.2.1 Algebraic Transformations

An algebraic transformation is of the form **; name = expression**. *Name* is the name of a variable. It may be an existing variable or a new variable. *Name* may have been read in or previously created.

The expression can be any algebraic transformation of any complexity. You may nest parentheses, functions, and operations to any level. Functions that may appear in expressions are listed in Section R4.3. The operators that may appear in **CREATE** commands are the standard ones, **+**, **-**, *****, and **/** for addition, subtraction, multiplication, and division, as well as the special operators listed below:

^	= raise to the power;	$a \wedge b$	$= a^b$
@	= Box-Cox transformation;	a @ b	$= (a^b - 1) / b$ or log$a$ if $b = 0$ and $a > 0$
!	= maximum;	$a \,!\, b$	$= \max(a,b)$
	(The maximum of a string of operands is obtained just by writing the set separated by !s. For example, 5 ! 3 ! 6 ! 0 ! 1 = 6.)		
~	= minimum;	$a \sim b$	$= \min(a,b)$
%	= percentage change;	$a \,\%\, b$	$= 100(a/b - 1)$ E.g., 5 % 4 = 25

The following operators create binary variables:

>	= binary variable;	*a*>*b*	= 1 if *a*>*b* and 0 else.
>=	= binary variable;	*a*>= *b*	= 1 if $a \geq b$ and 0 else.
<	= binary variable;	*a*<*b*	= 1 if *a*<*b* and 0 else.
<=	= binary variable;	*a*<= *b*	= 1 if $a \leq b$ and 0 else.
=	= binary variable;	*a* = *b*	= 1 if *a* = *b* and 0 else.
#	= binary variable;	*a* # *b*	= 1 if *a* is not equal to *b*.

For example,

CREATE	**; a = x > 0 * Phi(y)**	creates *a* equal to Phi(*y*) if *x* is positive and 0 else
	; p = z > 0	creates *p* = 1 if *z* is positive and 0 otherwise
	; zeq1 = z = 1 $	equals 1 if *z* equals 1 and 0 otherwise.

To avoid ambiguity, it is often useful to enclose these operations in parentheses, as in

```
CREATE        ; a = (x = 1) * Phi(z) $
```

This set of tools can be used in place of conditional commands, and sometimes provides a convenient way make conditional commands. For example 'and' conditions result from products of these relational operators. Thus,

```
CREATE        ; v = (x >= 8) * (x <= 15) * Log(q) $
```

creates *v* equal to the log of *q* if *x* is greater than or equal to 8 *and* less than or equal to 15. You can also produce an 'or' condition using addition, though the conditional command construction shown below may be more convenient. For example:

```
CREATE        ; v = ( ( (x = 8) + (x = 15) )  >  0 ) * Log(q) $
```

does the transformation if *x* equals 8 or 15.

The following algebraic order of precedence is used to evaluate expressions:

- First: functions, such as Log(.) are evaluated.
- Second: ^ and @, which have equal precedence are computed.
- Third: *, /, !, ~, %, > , >=, <, <=, =, # are computed.
- The special operators, !, %, etc. are evaluated from left to right with the same precedence as * and /. Thus, for example, y * x > 0 equals 1 if y*x is greater than 0 and equals 0 otherwise. It will usually be useful to use parentheses to avoid ambiguities in these calculations.
- Fourth: + and - (addition and subtraction) are computed.

NOTE: *LIMDEP* does *not* give the unary minus highest precedence. The expression **-x^2** evaluates to the negative of the square of *x* (which would be negative) not the square of negative *x* (which would be positive). This is the current standard in software, but it is not universal.

You may use as many levels of parentheses as necessary in order to group items in an expression or to change the order of evaluation. For example,

```
CREATE      ; ma = (pz + pz[-1] + pz[-2] + pz[-3]) / 4 $
```

computes a moving average of a current and three lagged values. Parentheses may also be nested to any depth.

```
CREATE      ; ratio = ((x + y)^2-(a + c)^2)^2/((a + x)*(c + y)) $
```

is a valid command which computes $ratio = \frac{((x+y)^2-(a+c)^2)^2}{(a+x)(c+y)}$.

NOTE: Implied multiplication of expressions in parentheses is allowed, but you should be very careful when you use this feature to avoid ambiguity. The use of the '*' to indicate multiplication will help to clarify exactly what the expression should be. Nonetheless, you can use products such as (a + x) (c + y) which will evaluate correctly. In the **CREATE** command above, the '*' in the denominator could be omitted, since with implied multiplication, the expression is correct without it.

You may also nest functions. For a few examples, consider the inverse probability distributions in the discussion of 'other distributions for survival models' in Chapter E60. The expressions shown there can be created exactly as they are listed. Thus,

```
Gompertz:    CREATE ; t  = Log(1 - w*Log(a)/p) / w $
Weibull:     CREATE ; t  = (-Log(a))^(1/p) / w $
Normal:      CREATE ; t  = Exp(-Inp(a)/p) / w $
Logistic:    CREATE ; t  = ((1 - a) / a ) ^ (1/p) / w $
```

Functions may be nested to any depth, and expressions may appear in the parentheses of a function. Consider, for example, the following which creates the terms in the log likelihood function for a tobit model

```
CREATE      ; loglik = (1 - d)  *  Log(Phi(-x'b/sigma))
                     + d  *  Log((1/sigma)*N01((y-x'b)/sigma)) $
```

Four Cautions:

- Any transformation that involves a missing value (-999) at any point returns a missing value.
- It is unlikely to be necessary, but if you should require expressions in the parameter list of a two parameter function, put them in parentheses. The Trn function which computes trend variables is such a function. Thus,

  ```
  CREATE   ; trend = Trn(a+b'x , step ) $
  ```

  would confuse the compiler. Instead, you should use

  ```
  CREATE   ; trend = Trn( (a+b'x) , step ) $
  ```

- In specifying lags, if the lag is an expression, for example, in a loop, enclose the expression in parentheses. Thus,

  ```
  CREATE   ; looplag = x [ i + 2*j ] $
  ```

  will not work, but, you could use

  ```
  CREATE   ; looplag = x [ (i + 2*j) ] $
  ```

- Many operations allow you to access particular observations of a variable by using an observation subscript enclosed in parentheses. If you will be using this construction, you must avoid variable names which are the same as the function names listed in Section R4.3. For example, if you have a variable named *phi*, then Phi(1) could be the first observation on *phi* or the standard normal CDF evaluated at 1.0. (**CREATE** will translate it as the latter.) Function names all have three letters.

Variables may appear on both sides of the equals sign as long as they already exist, and transformations may be grouped in a single command. In a multiple **CREATE** command, later transformations may make use of variables created in earlier ones. For example,

```
CREATE       ; sam = x1 * x2
             ; bob  = x2 + x3
             ; this  = sam * bob
             ; that = Log(this)
             ; that = 1 / that $
```

is the same as five consecutive **CREATE** commands. Grouping commands in this fashion is more efficient when it is possible to do so. Each **CREATE** command requires a pass through the data set. Thus, the preceding requires only a single pass, whereas if you were to write it as five separate commands, you would require five passes. For moderate sized samples (less than 10,000), this won't make much difference. But, if your sample size is huge, you will want to make use of this result. On the other hand, combining transformations, such as eliminating the first two commands and making *this* = (*x*1**x*2)*(*x*2+*x*3), within a single **CREATE** command generally does not save any time, as the same amount of computation must be done either way. For this consideration, you should write your transformations so that they are as 'self documenting' as possible – that is, so that they are as easy to understand as possible.

R4.2.2 Conditional Transformations

Any transformation may be made conditional. The essential format is

CREATE ; If (logical expression) name = expression $

Logical expressions are any desired expressions that provide the condition for the transformation to be carried out. They may include any number of levels of parentheses and may involve mathematical expressions of any complexity involving variables, lagged variables of the form **name[lag]**, named scalars, matrix or vector elements, and literal numbers. The operators are the same as above with a few exceptions: The ones that may be used are the math and relational operators: +, -, *, /, ^, >, >=, <, <=, =, #. The special operators, @, !, %, and ~ are not used here.

NOTE: Logical expressions may not involve functions such as Log, Exp, etc.

Concatenation operators which can be used for transformations are & for 'and,' and | for 'or.' A simple example might be: **CREATE ; If (x > 0) ... expression $** For a more complex example, we compute an expression for observations which are not inside a ball of unit radius.

CREATE ; If (x1^2 + x2^2 + x2^2 >= 1) ...expression... $

For a third example with no obvious interpretation:

CREATE ; If ((r/s)*((c+7)*(x+2) * y^2 + z^3) > 1 | x+y < 0) ...expression... $

The hierarchy of operations is ^, (*, /) (+,-), (>,>=,<,<=,=,#), &, |. Operators in parentheses have equal precedence and are evaluated from left to right. When in doubt, add parentheses. There is essentially no limit to the number of levels of parentheses. (They can be nested to about 20 levels.)

Comparisons to the Missing Value Code

Although you may not transform missing values in algebraic expressions, you may base comparisons on them. Thus, you may use **If (name = -999)...** to base a computation on missing data. Since it is also possible to change the missing value, a more convenient form would be to use the "." standard; thus **If (name = .)…** will also work,

If / Else Transformations

An '**If/Else**' construction may also be specified as follows:

CREATE ; If (...) name = expression; (Else) name = expression $

The condition is tested first. If it is false and '*name*' does not already exist, a value of 0.0 is returned for the expression. If it is false and *name* does already exist, then the current value is left unchanged for that observation. If it is true, the expression is evaluated and its actual value is returned. In a succeeding **(Else)**, if the preceding **If (...)** was false, the expression is computed. Any valid **CREATE** expression may appear in either place; the second (after the **(Else)**) may, if desired, be unrelated to the first. For example:

CREATE ; If (age > 21 & ftjob = 1) adult = 1 ; (Else) child = 1 $

Conditions Applied to Groups of Transformations

A condition may be applied to a group of commands by using

```
CREATE    ; If (condition)   | a set of transformations $
```

An alternative set of transformations may also be computed using **(Else)**, as follows:

```
CREATE    ; If (condition)   | a set of transformations
          ;    (Else)        | a different set of transformations $
```

The second set of transformations need not be related to the first, though it could be. For example,

```
CREATE    ; If (x = 1)       |  z1 = Log(z1) ; z2 = Log(z2)
          ;    (Else)        |  z1 = Exp(z1) ; z2 = Exp(z2) ; z4 = z1*z2 $
```

Note that *z*4 is only computed if *x* is not equal to 1. The value given to *z*4 when *x* equals 1 depends on whether *z*4 existed prior to the command; if yes, it is unchanged, if no, it equals 0. This is *LIMDEP*'s usual convention. Finally, you may switch off the batch control with

```
CREATE    ; If (condition)   | a set of transformations
          ;    (Else)        | a different set of transformations
          ;    (Endif)       | more transformations $
```

Those transformations which follow the **Endif** are always carried out. The **If/Else** conditions are no longer controlling. Note that this is the same as

```
CREATE    ; If (condition)   | a set of transformations
          ;    (Else)        | a different set of transformations $
CREATE    ; more transformations  $
```

The reason for using the first construction instead of the second is for speed. The first construction requires one pass through your data while the second requires two. If you have a small sample, you will not notice the difference. But, if you have tens of thousands of observations, the first form of the two commands might bring a noticeable time savings.

WARNING: You can have conditional transformations under the control of these **If/Else** setups in a command. For example: **CREATE ; If (x = 1) | z = 3 ; If (y > 0) y = Log(y) $**. But, this probably will not produce the desired results, since the second condition will not be tested if the first fails. Also, using **(Else)** while inside an **(Else)|** block will produce unpredictable results, and should be avoided. It is better to break up the **CREATE** command into more than one command.

Logical Expressions

It is important to note that in evaluating expressions, you get a logical result, not a mathematical one. The result is either true or false. An expression which cannot be computed cannot be true, so it is false. Therefore, any subexpression which involves missing data or division by zero or a negative number to a noninteger power produces a result of false. But, that does not mean that the full expression is false. For example: **If (x/0 > 0 | x > y) expression $** could be true. The first expression is false because of the zero divide, but the second might be true, and the *or* in the middle returns true if either expression is true. Also, we adopt the C++ language convention for evaluation of the truth of a mathematical expression. A nonzero result is true, a zero result is false. Thus, your expression need not actually make logical comparisons. For example: Suppose *x* is a binary variable (zeros and ones). **CREATE ; If (x) expression $** will compute the expression for observations for which *x* equals one and not compute it when *x* equals zero, since the expression has a value of 'true' when *x* is not zero. Therefore, this is the same as **CREATE ; If (x # 0) expression $**.

This syntax produces vast flexibility. However, there is one possible ambiguity as a result. Numbers in exponential format *must* be in the form '**snnnnn.D+ee**' or '**snnnnn.E+ee**,' where 's' may be a minus sign (do not include superfluous '+' signs), and 'ee' is a one or two digit exponent. I.e., although 1.2D+2 and 12.D+1 are the same number, the first will produce an unexpected result – use the second. The first form might produce a syntax error, depending on the rest of the command, but more likely, would just produce a result that was not calculated the way you expect.

Making the Entire Command Conditional

A way to make the entire **CREATE** command conditional is

```
CREATE     ; If [condition as usual] | a set of transformations $
```

Note the use of square brackets. The condition is tested. If it is true, all of the following transformations are carried out. If false, none are. For example,

```
CREATE     ; If [j = 1] | z1 = Log(z1) ; z2 = Log(z2) $
```

The two transformations are computed if *j* equals 1. If not, neither is carried out. The difference between this form and

```
CREATE     ; If (condition as usual) | a set of transformations $
```

is that the form with parentheses is evaluated during a data loop. The transformation is evaluated for each observation. It might be carried out for some observations and not for others. For example,

```
CREATE     ; If (Sex = Male) | a set of transformations $
```

will be carried for some observations and not for others. But, in the form with square brackets, the condition is evaluated before anything else is done in the transformation program. If the condition is false, the entire **CREATE** command that follows is ignored. Thus, you might use

```
REGRESS    ; Lhs = y ; Rhs = x $
CREATE     ; If [sumsqdev > 100] | a set of transformations $
```

Note that you would not want to use a variable in such a condition, though it would not cause problems for the command processor – the condition is only evaluated once, so the result would be unpredictable.

R4.3 CREATE Functions

The expressions in **CREATE** may involve the following functions:

R4.3.1 Common Algebraic Functions

Log(*x*)	= natural logarithm
Exp(*x*)	= exponent
Abs(*x*)	= absolute value
Sqr(*x*)	= square root
Sin(*x*)	= sine
Cos(*x*)	= cosine
Tan(*x*)	= tangent
Rsn(*x*)	= arcsine (operand between -1 and 1)
Rcs(*x*)	= arccosine (operand between -1 and 1)
Ath(*x*)	= hyperbolic arctangent = ½Log((1+*x*)/(1-*x*)), $-1 < x < 1$
At1(*x*)	= $1/(1 - x^2)$
Ati(*x*)	= inverse hyperbolic arctangent = [Exp(2*x*)-1]/[Exp(2*x*)+1]
Hsn(*x*)	= hyperbolic sin = .5 (Exp(2*x*)-1)/Exp(*x*)
Hs1(*x*)	= dHsn(*x*)/d*x* = Hcs(*x*)
Hcs(*x*)	= hyperbolic cos = .5*(Exp(2*x*)+1)/Exp(*x*)
Hc1(*x*)	= dHcs(*x*)/d*x* = Hsn(*x*)
Htn(*x*)	= hyperbolic tangent = Hsn(*x*)/Hcs(*x*)
Ht1(*x*)	= dHtn(*x*)/d*x*) = 1 / Hcs$(x)^2$
Gma(*x*)	= gamma function = (*x*-1)! if *x* is a positive integer
Psi(*x*)	= digamma = log-derivative of gamma function = $\Gamma'/\Gamma = \Psi(x)$
Psp(*x*)	= trigamma = log-2nd derivative of gamma = $(\Gamma\Gamma''-\Gamma'^2)/\Gamma^2 = \Psi'(x)$
Lgm(*x*)	= log of gamma function (returned for Gma if $x > 50$)
Bta(x,y)	= complete beta function, x,y > 0
Sgn(*x*)	= sign function = -1,0,1 for *x* <, =, > 0
Fix(*x*)	= round to nearest integer
Int(*x*)	= integer part of operand
Min(*x*,*y*)	= minimum of *x* and *y*
Max(*x*,*y*)	= maximum of *x* and *y*
Mxx(*x*)	= maximum value of variable *x* in current sample
Mnx(*x*)	= minimum value of variable *x* in current sample

R4.3.2 Univariate Normal and Related Distributions

Phi(*x*)	= CDF of standard normal
N01(*x*)	= PDF of standard normal
Lgf(*x*)	= log of standard normal PDF = -.5(Log2π + x^2) = Log(N01(*x*))
Lmm(*x*)	= -N01/Phi = E[*x* | *x* < operand], *x* ~ N(0,1)
Lmp(*x*)	= N01/(1-Phi) = E[*x* | *x* > operand]
Lmd(*x*,*z*)	= (*z*-1)Lmp(*x*) - *z*Lmm(*x*) where *z* = 0/1 (selectivity variable)
Tvm(*x*)	= [1 - Lmm(Lmm+z)] = Var[*x* | *x*< operand]
Tvp(*x*)	= [1 - Lmp(Lmp+z)] = Var[*x* | *x*> operand]
Tvr(*x*,*z*)	= (1-*z*)Tvm(*x*) + *z*Tvp(*x*) where *z* = 0/1 (selected variance)
Inp(*x*)	= inverse normal CDF
Inf(*x*)	= inverse normal PDF (operand is CDF, returns density)
Erf(*x*)	= error function = 2Φ(Sqr(2)*x*) – 1
Erc(*x*)	= complementary error function = 2(1 - Φ(Sqr(2)*x*))
Tdd(*x*,*d*)	= t density with *d* degrees of freedom at *x*
Tdf(*x*,*d*)	= t CDF with *d* degrees of freedom at *x*
Cdd(*x*,*d*)	= chi squared density with *d* degrees of freedom at *x*
Cdf(*x*,*d*)	= chi squared CDF with *d* degrees of freedom at *x*

R4.3.3 Logistic and Discrete Distributions

Lgt(*x*)	= logit = Log[*z*/(1-z)]
Lgp(*x*)	= logistic CDF = Exp(*x*)/(1 + Exp(*x*))
Lgd(*x*)	= logistic density = Lgp(1-Lgp)
Psn(x,λ)	= CDF of Poisson distribution
Psd(x,λ)	= Poisson probability
Nbp(x,λ,q)	= negative binomial CDF
Nbd(x,λ,q)	= negative binomial probability
Bnm(x,π,K)	= binomial CDF
Bnd(x,π,K)	= binomial probability

R4.3.4 Trends and Seasonal Dummy Variables

Trn(*x*1,*x*2) = trend = *x*1+(*i*-1) *x*2 where *i* = observation number.

There are two forms of the Trn function that are useful for panel data.

Trn(*T*,0) = 1,1,…,2,2,…,3,3,…,*N*,*N*,…

where each block repeats the sequence number *T* times. The function

Trn(-*T*,0) = 1,2,…,*T*, 1,2,…,*T*, 1,2,…,*T* …

Each of the *N* blocks in the data contains a sequence of integers from 1 to *T*.

Ind($i1$,$i2$)	= 1 if $i1 \le$ observation number $\le i2$, 0 else,
Dmy(p,$i1$)	= 1 for each pth observation beginning with $i1$, 0 else.

The Dmy function is used to create seasonal dummy variables. The Ind function operates on specific observations, as in

CREATE ; eighties = Ind(22,31) $

If your data are time series and have been identified as such with the **DATES** command (see Chapter R7), then you may use dates instead of observation numbers in the Ind function, as in

CREATE ; eighties = Ind(1980.1,1989.4) $

NOTE: The Ind function is oblivious to centuries. You must provide four digit years to this function, so there is no ambiguity about 19xx vs. 20xx.

The trend function, Trn is used to create equally spaced sequences of values, such as 1,2,3,..., which is Trn(1,1). There are two additional variants used primarily with panel data. These are discussed in Chapter R5.

R4.3.5 Ranks of Observations

Rnk(x)	= ranks of sorted x.

For example, if the current sample of x contains values 8,2,0,3,1, then the transformed variable

CREATE ; Ranks = Rnk(x) $

creates a variable which equals 5,3,1,4,2.

R4.3.6 Box-Cox Function and its Derivatives

The Box-Cox function is $x@c = (a^c - 1)/b$ or log$a$ if $c = 0$. The derivatives of this function obey the differential equation

$$d^i(x@c)/dc^i = (1/c)(x^c(\log x)^i - i\,d^{\,i-1}(x@c)/dc^{i-1}) \text{ or } (\log x)^{i+1}/(i+1) \text{ if } c = 0.$$

The functions Bx1(x,c) and Bx2(x,c) may be used to obtain the first and second derivatives.

R4.3.7 Bivariate and Multivariate Normal Probabilities

You may obtain bivariate normal probabilities using the following construction:

Bvn(*x*1.*x*2, *r*)	= bivariate normal CDF,
Bvd(*x*1.*x*2, *r*)	= bivariate normal density,
Bv1(*x*1.*x*2, *r*)	= partial derivative of Φ_2 (*z*1, *z*2, ρ) with respect to *z*1,
Bv2(*x*1.*x*2, *r*)	= partial derivative of Φ_2 (*z*1, *z*2, ρ) with respect to *z*2.

Previous versions of *LIMDEP* required the *x*1, *x*2 pair to be in a namelist. That syntax may still be used – all commands are forward compatible. However, this new form allows *x*1 and/or *x*2 to be numbers or scalars instead of variables, which the old one did not. Note that *r* need not be a number; it may also be a variable and vary by observation. For example, the following replicates the probabilities computed by a bivariate probit model:

```
CREATE      ; q1 = 2*y1-1 ; q2 = 2*y2-1 $
CREATE      ; bivprob = Bvn((q1*b1'x1), (q2*b2'x2, (q1*q2*rho)) $
```

(If you need this function for scalars instead of variables, use **CALC**.)

HOW IT'S DONE: See the appendix to this chapter for details on this computation.

For multivariate normal probabilities, use

```
CREATE      ; prob = Mvn(x,w) $
```

in which *x* is a namelist of *M* variables. Each row (observation) in namelist (matrix) *x* is the counterpart to the *x* in the **CALCULATE** function. The namelist, *x* includes *M* variables and the matrix, *w* is the *M*×*M* covariance matrix. Note, variables may be repeated in *x*. For example, if *x*1 and *x*2 are free, but *x*3 - *x*6 are all 0, then you could use

```
CREATE      ; zero = 0 $
NAMELIST    ; x = x1,x2,zero,zero,zero $
CREATE      ; p = Mvn(x,w) $
```

This creates a variable *p* with each element equal to the *M*-variate normal CDF evaluated at *w* and the *i*th observation in *p*. The Mvn function may be used as you would any other function in **CREATE**. The function Mvd(*x*,*w*) returns the density instead of the CDF.

HOW IT'S DONE: See the appendix to this chapter for details on this computation.

R4.3.8 Leads and Lags

You can use a lagged or leaded variable with the operand

variable [*n*] = observation on the variable *n* periods prior or ahead.

The use of square brackets is mandatory; '*n*' is the desired lag or lead. If *n* is negative, the variable is lagged; if it is positive, it is leaded. For example, Nerlove's 'universal filter' is $(1 - .75L)^2$ where *L* is the lag operator. This would be

```
CREATE      ; filterx = x - 1.5 * x[-1] + .5625 * x[-2] $
```

A value of -999 is returned for the operand whenever the value would be out of the range of the current sample. For example, in the above command, *filterx* would equal -999 for the first two observations. You can change this default value to something else, like zero, with

```
CREATE      ; [lag] = the desired value  $
```

For example,

```
CREATE      ; [lag] = 0 $
```

would change the default value for noncomputable lags to zero. This must be used in isolation, not as part of some other command.

If you use lags or leads, you should modify the applicable sample accordingly when you use the data for estimation. *LIMDEP* makes no internal note of the fact that one variable is a lagged value of another one. It just fills the missing values at the beginning of the sample with -999s at the time it is created.

Moving average and autoregressive sequences can easily be constructed using **CREATE**, but you must be careful to set up the initial conditions and the rest of the sequence separately. Also, remember that **CREATE** does not reach beyond the current sample to get observations. A special read-only variable named *_obsno* (note the leading underscore) is provided for creating recursions. Consider computing the (infinite) moving average series

$$y_t = x_t + \theta x_{t-1} + \theta^2 x_{t-2} + ... + \theta^{t-1} x_{t-1.}$$

To do the computation, we would use the autoregressive form, $y_t = x_t + \theta y_{t-1}$ with $y_1 = 0$. The following could be used:

```
CREATE      ; If (_obsno = 1) y = 0
            ;    (Else)        y = x + theta * y[-1] $
```

Second, consider generating a random sample from the sequence $y_t = \theta y_{t-1} + e_t$, where e_t ~N[0,1]. Simply using **CREATE ; y = theta*y[-1] + Rnn(0,1) $** will not work, since, once again, the sequence must be started somewhere. But, you could use the following

```
CREATE      ; If (_obsno = 1) y = Rnn(0,1) / Sqr(1 - theta^2)
            ;    (Else)        y = theta * y[-1] + Rnn(0,1) $
```

R4.3.9 Matrix Functions

Two transformations based on matrix algebra are used to create linear and quadratic forms with the data. Linear combinations of variables are obtained with

```
CREATE      ; name = b'x $
```

where *x* is a namelist of variables (see Chapter R6) and *b* is any vector with the same number of elements. It creates the vector of values from the linear combination of the variables in the namelist with coefficients in the row or column matrix. Dot products may also be used with other transformations. For example,

```
CREATE      ; bx12 = x1'b1 / x2'b2 ; p = Phi(x'b/s) $
```

(The order is not mandatory. **d'z** is the same as **z'd**.) Also, if you need this construction, a dot product may be used for two vectors or two namelists. In the latter case, the result is the sum of squares of the variables. (See the Xmt function described in Section R4.3.13.)

Quadratic forms are computed with the Qfr function:

```
            ; Qfr (namelist, matrix)
```

This computes $q = \mathbf{x}'\mathbf{A}\mathbf{x}$ where **x** is a column vector and **A** is a matrix. A matrix function Qfrm is available for this computation for a single vector. There are occasions when you might wish to obtain this result for each row in a set of data. For example, if you have a vector of parameters, **b**, and an estimated covariance matrix for them, **V**, you might compute, for each observation in a data set, a fitted value, $yfit_i = \mathbf{b}'\mathbf{x}_i$. To obtain a standard error for each of these values, you would require s_{fi} = **Sqr**[$\mathbf{x}_i'\mathbf{V}\mathbf{x}_i$] for each observation, *i*. This can be obtained with the Qfr function in **CREATE**. To use this function, you must first define the namelist containing the names of the variables in $\mathbf{x}_i$. The matrix must be square with number of rows equal to the number of variables. The command is then as shown above. For example, to obtain the fitted values and forecast standard errors for the most recent regression, you might use

```
NAMELIST    ; x = one,gnp,prices $
REGRESS     ; Lhs = cons ; Rhs = x $
CREATE      ; cfit  = x'b; scfit = Sqr(s^2 + Qfr(x,varb)) $
```

This uses two matrices and a scalar automatically saved by the regression command. For another example, in linear regression analysis, the 'hat' matrix,

$$\mathbf{H} = \mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'.$$

is useful for computing regression diagnostics. The matrix **H** is *n* x *n*, which might be huge – *n* could be hundreds of thousands – but typically, only the diagonal elements are useful. The following can be used to obtain *hii*, the diagonal elements, in a variable:

```
NAMELIST    ; x = the list of variables $
MATRIX      ; xxi = <x'x> $
CREATE      ; hii = 1 - Qfr(x,xxi) $
```

R4.3.10 Moving a Matrix

There are occasions when you want to move a matrix computed with the **MATRIX** command to a place where you can manipulate it as if it were a set of data, instead. Normally, you can do all of this with **MATRIX**, but it might be useful to change a 'matrix' into a 'variable.' (The necessary distinction is discussed in Chapter R16.) For an example, when you fit a fixed effects model with **REGRESS**, the vector of group specific constants is saved as a matrix. You might be interested in using these fixed effects as a variable, for example, in computing regressions with them as observations on the 'dependent variable.'

You can move a vector to a variable just by equating them. To continue our example,

```
REGRESS     ; ... ; Panel ; Fixed Effects $
CREATE      ; va =  alphafe  $
```

does the required transformation. You can also move a matrix to a namelist, column by column. Use

```
CREATE      ; x = a $
```

where *x* is defined by a namelist and *a* is a matrix which has number of columns equal to the number of variables in *x*. (The namelist may not contain *one*. See Chapter R8 for discussion of using *one* as the constant term in a model.) You may move a matrix to a variable, with **CREATE ; z = a $** which moves the first column of matrix *a* to variable *z*. You can also copy one namelist in to another. Use

```
CREATE      ; y = x $
```

to copy all variables in *x* into *y* (as many as possible if they have different numbers of columns).

R4.3.11 Means, Deviations, Standardized Variables

The matrix functions Mean, Xdev and Xstd are used to obtain sample means, to center (subtract the mean) and standardize a set of variables listed in a data matrix (namelist). For some purposes, and if you are using just a single variable, it may be more convenient to use the **CREATE** command directly to operate on the variable. The functions Xbr, Dev, and Std are provided for this purpose. Thus, **CREATE ; y = Dev(x) $** creates the variable *y* by subtracting *x*'s mean from each observation. An equivalent command would be **CREATE ; y = x - Xbr(x) $**.Likewise, the means of both *y* and *z* in

```
CREATE      ; y = x * Dev(x) ; z = Dev(x) ^ 2 $
```

are equal to the variance of *x*. Standardized data are obtained with the Std function. For example, after the following: **CREATE ; y = Std(x) ; w = Std(x) * Std(q) $** *y* has mean 0 and standard deviation 1 while the mean of the variable *w* is the correlation between *x* and *q*. (There is an easier way to compute this.)

HOW IT'S DONE: See Appendix R4A.8 to this chapter for details on the computation of sample variances.

The standard deviation of a variable, defined with Sdv(*x*), may be used in other functions in **CREATE**. For example, to standardize a variable (the hard way)

```
CREATE      ; stdx = (x - Xbr(x)) / Sdv(x) $
```

R4.3.12 Regression Residuals and Fitted Values

Some calculations use a vector of residuals or predictions from a linear regression as a side result in some other calculation. For example, one way to compute the two stage least squares coefficients when there is an endogenous variable in a model is to add the residuals from the regression of the endogenous variable on the instruments to the original equation and use least squares. The Res and Fit functions can be used for that calculation. For example,

```
NAMELIST    ; z = the full set of exogenous variables $
NAMELIST    ; x = the Rhs of the model,xendog $
CREATE      ; uxe = Res(xendog,z) $
REGRESS     ; Lhs = y ; Rhs = x,uxe $
```

(Of course, you could just use 2SLS, but this shows how to use this function.) The forms of the two functions are

Res (variable, namelist) and **Fit (variable, namelist)**.

Note, the regression is computed using the current sample.

R4.3.13 Moments for a Set of Variables – the Xmt Function

All of the preceding discussion describes operations on the set of observations on a variable. Another possibility is operation on a set of variables. (The dot product operation and Qfr function are such operations.) If namelist *x* is a set of at least two variables, the **CREATE** function

Xmt(*x*,*j*) = *j*th moment of the set of observations defined by a row of namelist *x*,

computes a statistic for the *K* variables, once for each observation. '*x*' must be the name of a namelist, not a variable and not a set of variables; *J* must be an integer ranging from one to ten. For

j = 1,	Xmt(..)	= mean of variables
j = 2,	Xmt(..)	= standard deviation
j = 3,...,10,	Xmt(..)	= centered and scaled moment.

For example, Xmt(*x*,3) computes the standardized skewness measure

$$\text{Xmt}(x,3)_i = (1/K)\Sigma_{k=variables}\,[x_{ik} - \bar{x}_i,]^3/s_i^3,\ i = \text{observations } 1,\ldots,N,$$

where $\bar{x}_i$ is the mean and s_i is the standard deviation of the set of *K* variables at observation *i*. Powers up to 10 are available (the interpretation is left to the user). Note that this is computed for each observation.

For example, if *x* contains 10 variables, e.g., test scores, and the sample contains 50,000 observations, the sum is over the 10 variables, for each observation. For example, for a sample of 175,000 observations on a battery of tests,

```
NAMELIST    ; tests = math,reading,physics,history,algebra,golf $
CREATE      ; skewness = Xmt(tests,3)
            ; kurtosis = Xmt(tests,4) $
```

computes two new variables in the data set; *skewness* is the sample of 175,000 observations on the skewness of the sample of six test scores for each observation while *kurtosis* is the kurtosis.

The sum across variables of a set of variables can be obtained with

```
CREATE      ; name = x'1 $  or  1'x
```

where *x* is a namelist and 1 is the literal, number one. Note that this sums across variables, not observations. Each observation is the sum of the variables for that observation.

R4.3.14 Multiple of a Set of Variables – the Scl Function

If *x* is a set of variables defined by a namelist and *v* is a variable, the command

```
CREATE      ; newx = Scl(x,v) $
```

creates a replica of the entire set of variables in *x* with each observation on each variable in *x* multiplied by the corresponding observation of variable *v*.

NOTE: This function creates a new namelist and a new set of variables.

The function operates as follows: The new namelist is given the name that appears on the Lhs of the equation. The new variables created have names constructed from the namelist name by appending the number of the variable. These variables may already exist, in which case they are just overwritten. A simple example would be as follows: (We consider a more substantive example later.)

```
CREATE      ; x1 = 2 ; x2 = 3 ; ten = 10 $
NAMELIST    ; x = x1,x2 $
CREATE      ; tenx = Scl(x,ten) $
```

The first command creates three variables, *x*1, *x*2, and *ten* equal to 2, 3, and 10, respectively, at every observation. The Scl function creates two new variables, *tenx*1, and *tenx*2, which equal 20 and 30, respectively, at every observation. It also creates a namelist named *tenx* which contains *tenx*1 and *tenx*2. The last command is equivalent to

```
CREATE      ; tenx1 = ten * x1
            ; tenx2 = ten * x2 $
NAMELIST    ; tenx = tenx1,tenx2 $
```

TIP: A namelist can contain up to 150 names, so this function can combine a very large number of commands.

NOTE: The name for the namelist in the Scl function must have six or fewer characters.

The Scl transformation must be the only one on the command line. Any other transformations will be ignored. For example, if the first and third commands in the example above were combined in

```
CREATE      ; ten = 10 ; tenx = Scl(x,ten) $
```

The first transformation would not be carried out.

Scl could be used for setting up some specification tests which require the derivatives of a log likelihood. But, there is a more efficient way of doing this. See Chapter R16 for an extensive application.

HINT: This command can produce a huge amount of data and can easily exhaust your data array if *x* has many variables. Use carefully. This should rarely if ever be necessary. In most situations in which you would use this function, what you will actually need is a moment matrix built up from **newx′newx**. There will always be a more efficient way to obtain this result than actually replicating the data matrix, *x*.

R4.3.15 Expanding a Categorical Variable into a Set of Dummy Variables

It is often useful to transform a categorical variable into a set of dummy variables For example, a variable, *educ*, might take values 1, 2, 3, and 4, for less than high school, high school, college, post graduate. For purposes of specifying a model based on this variable, one would normally expand it into four dummy variables, say *underhs*, *hs*, *college, postgrad*. This can easily be done with a set of **CREATE** commands, involving, for example, *hs* = (*educ* = 2) and so on. *LIMDEP* provides a single function for this purpose that simplifies the process and also provides some additional flexibility. The categorical variable is assumed to take values 1,2,...,*C*.

The command

CREATE ; Expand(variable) = name for category 1, ... name for category C $

does the following:

- A new dummy variable is created for each category. (If the variable to be created already exists, it is overwritten).
- A namelist is created which contains the names of the new variables. The name for the namelist is formed by appending an underscore both before and after up to six characters of the original name of the variable.
- A tabulation of the transformation is produced in the output window.

The example suggested earlier might be simulated as follows, where the commands and the resulting output are both shown:

CREATE ; educ = Rnd(4) $
CREATE ; Expand(educ) = underhs,hs,college,postgrad $

```
EDUC     was expanded as _EDUC_  .
Largest value =   4.   4 New variables were created.
Category
1     New variable = UNDERHS     Frequency=      28
2     New variable = HS          Frequency=      22
3     New variable = COLLEGE     Frequency=      30
4     New variable = POSTGRAD    Frequency=      20
Note, this is a complete set of dummy variables.  If
you use this set in a regression, drop the constant.
```

As noted, the transformation begins with the value 1. Values below 1 are not transformed and no new variable is created for the missing category. Also, the transformation does not collapse or compress the variable. If you have empty categories in the valid range of values, the variable will simply always take the value 0.0. Thus, if *educ* had been coded 2, 4, 6, 8, then the results of the transformation might have appeared as shown below

```
EDUC     was expanded as _EDUC_  .
Largest value =   8.   4 New variables were created.
Category
1     New variable = UNDERHS     Frequency=       0  <--- !
2     New variable = HS          Frequency=      23
3     New variable = COLLEGE     Frequency=       0  <--- !
4     New variable = POSTGRAD    Frequency=      29
5     New variable = EDUC05      Frequency=       0  <--- !
6     New variable = EDUC06      Frequency=      22
7     New variable = EDUC07      Frequency=       0  <--- !
8     New variable = EDUC08      Frequency=      26
Note, this is a complete set of dummy variables.  If
you use this set in a regression, drop the constant.
```

The Expand specification works as follows:

- The empty cells are flagged in the listing, but the variable is created anyway.
- If your list of names is not long enough, the remaining names are built up from the original variable name and the category value.
- The program warns you that this has computed a complete set of dummy variables. If you use this set of variables in a regression or other model, you should not include an overall constant term in the model because that would cause perfect collinearity – the ‘dummy variable trap.’ Thus, a model which contained both *one* and *_educ_* would contain five variables that are perfectly collinear.

You may want to avoid the last of these without having to choose one of the variables to omit from the set. You can direct the transformation to drop one of the categories by adding ‘**,0**’ after the variable name in the parentheses.

CREATE ; Expand(variable,0) = list of names $

For our previous example, this modification would change the results as follows:

CREATE ; Expand(educ,0) = underhs,hs,college,postgrad $

```
EDUC     was expanded as _EDUC_  .
Largest value =   4.   0 New variables were created.
Category
1     New variable = UNDERHS     Frequency=      27
2     New variable = HS          Frequency=      26
3     New variable = COLLEGE     Frequency=      21
Note, the last category was not expanded. You may use
this namelist as is in a regression with a constant.
```

The note at the end of the listing reminds you of the calculations done. The last category is the one dropped. (Note that '0 new variables were created.' The reason is that these variables already existed after our earlier example.)

Finally, the list of names for the new variables is optional. If it is omitted, names are built up as in the second example above. Continuing the example, we might have

CREATE ; educ = Rnd(4) $
CREATE ; Expand(educ) $

```
EDUC     was expanded as _EDUC_  .
Largest value =   4.   4 New variables were created.
Category
1     New variable = EDUC01      Frequency=      28
2     New variable = EDUC02      Frequency=      22
3     New variable = EDUC03      Frequency=      30
4     New variable = EDUC04      Frequency=      20
Note, this is a complete set of dummy variables.  If
you use this set in a regression, drop the constant.
```

NOTE: This transformation will refuse to create more than 100 variables. If it reaches this limit, you have probably tried to transform the wrong variable. Thus, the variable must be coded 1,2,..., up to 99.

NOTE: This function for **CREATE** actually creates the set of dummy variables and the namelist associated with them. Your main use of categorical variables will be in specifying models with categorical variables. You will often use this computation without actually needing the dummy variables in the data set or the namelist. *LIMDEP* provides a way to do this directly in a command. To consider an example, (based on the health care data used in several examples in Greene (2012), consider a probit model which contains a variable *hsat* (health satisfaction) coded 1 to 11. The following three commands will produce the identical results:

CREATE ; Expand(hsat) $
PROBIT ; Lhs = public ; Rhs = one,age,educ,_hsat_ $

PROBIT ; Lhs = public ; Rhs = one,age,educ,Expand(hsat) $

PROBIT ; Lhs = public ; Rhs = one,age,educ,# hsat $

The third shows an abbreviation that can be used for the second. The second and third differ from the first in that they do not add variables to the data set. The inline expansion of categorical variables in a model command is shown in Chapter R8.

R4.3.16 Stacking Data to Create Data Matrices

The stacking operation is used to create specific kinds of data matrices. Consider an example – this would be used to set up generalized least squares of a particular regression model. The sample contains *N* observations on y1, y2, y3, x1, x2, x3, x4. We need a data matrix (data set) with 3*N* observations that appears as

$$\mathbf{y} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \end{bmatrix}, \ \mathbf{X} = \begin{bmatrix} 1 & \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \mathbf{x}_4 \\ 1 & \mathbf{x}_2 & \mathbf{0} & \mathbf{0} & \mathbf{x}_4 \\ 1 & \mathbf{x}_3 & \mathbf{0} & \mathbf{0} & \mathbf{x}_4 \end{bmatrix}.$$

The commands that can be used to create these are

CREATE ; y = Stk(y1/y2/y3) $
CREATE ; x = Stk(1,x1,x2,x3,x4 / 1,x2,0,0,x4 / 1,x3,0,0,x4) $

Note that the stacking operations replicates a number *N* times, or otherwise stacks *N* observations.
The general form of the function is

Stk(row definition / row definition / …)

The operation creates *N* observations for each row definition, so when done, the sample will contain *N***N* rows observations. Within a row, you may have variables, namelists and scalars, either numbers or named scalars created by **CALC** or any other means. The row definitions must all contain the same number of entities. They may be a mixture of numbers, namelists and scalars. For example, in the example above, if the definition

NAMELIST ; x = x1,x2,x3,x4 $

had already been declared, then the stacking definition given earlier could be

CREATE ; x = Stk(1,x / 1,x2,0,0,x4 / 1,x3,0,0,x4) $

R4.3.17 Group Functions for Panel Data

To create a new variable that replicates for each observation in a group the mean of that group, use the group means function,

CREATE ; y = Group Mean (x, Str = name or number)
or **CREATE ; y = Group Mean (x, Pds = name or number)**

The function requires a panel data specification, the same sort as used to specify panels in the model commands. This is discussed in Chapter R5. This function produces a report when computed, such as:

```
Computed Variable Y  Group means  of INVC
Number of groups found in current sample was  210
Max group =     4, Min =     4, Average =    4.0
```

This function must be used in isolation, not as part of another command nor in a compound function. Use a new **CREATE** command for each variable. Other available panel data functions are

Group Sums (x, Str = spec or Pds = spec)	= group sum within group
Group Prod (x, Str = spec or Pds = spec)	= group product within group
Group Devs (deviations from own group means)	= group deviations
Group Lags (first observation becomes missing)	= group lagged value
Group Diff (first observation becomes missing)	= group first difference
Group Obs1 (x, Str = spec or Pds = spec)	= first observation in group
Group Obs2 (…) – Group Obs9 (…)	= second … 9th observation
Group Last (…)	= last observation in group
Group Chg1 (…) … Group Chg9 (…)	= difference from *j*th obs. (E.g., for Chg1, returns $x(i,t) - x(i,1)$
Group Xmax (…)	= largest value in the group
Group Tmax (…)	= the index of the largest value (E.g., if $x(i,3)$ is largest value, returns 3.)
Group Xmin (…) and Group (Tmin)	= minimum and index of min

For the transformations that use the group means, Group Mean and Group Devs, you may provide a set of weights (in square brackets) to be used within the group, as in

```
CREATE     ; xbar = Group Mean (x [weight variable], Pds = panel spec) $
```

Weights are scaled to sum to the group size.

In these functions, you can do the calculations for a set of variables contained in a namelist. To do this, you must create the empty columns first and declare them in a namelist. The new namelist and the one being transformed must have the same numbers of variables. For example,

```
CREATE     ; xb1,xb2,xb3 $  (creates three empty variables)
NAMELIST   ; xbar = xb1,xb2,xb3 ; x = x1,x2,x3 $ (assuming x1, x2, x3 all exist)
CREATE     ; xbar = Group Mean (x, Pds = ti) $
```

Three functions are provided for configuring a panel data set.

Group Size (panel id)	= group sizes
Group Time (panel id)	= internal period indicator
Group Nmbr (count variable)	= sequential group number

Group Size (id) works on any unique identifier within the panel, such as a person id, to create a variable that contains, within the group, the number of observations in the group. For example, suppose the panel contains two groups, one with three observations and one with two, and, initially, variables *personid*, *x*1 and *x*2.

personid	x1	x2	ti	time
1	3	13	3	1
1	9	22	3	2
1	8	14	3	3
2	4	9	2	1
2	0	11	2	2

The command **CREATE ; ti = Group Size (personid) $** would create the variable *ti* shown above. In nearly all of the applications of panel data in *LIMDEP*, the specific time period in the group is not used – in some settings, this is labeled the ‘exchangeable’ case. Thus, the ‘time’ variable is only a within group index of the order of the observations. There could be cases in which the actual time variable might be of use. For example, one of the data sets we will use in some examples in this manual is a panel in which the observations take place in 1984, 1985, 1986, 1987, 1988, 1991 and 1994. There are seven years of data, but they are not evenly spaced. For a group that contains all seven years of data (not all do), the Group Time function would create time = (1,2,3,4,5,6,7). In actual time, the observations are (1,2,3,4,5,8,11). Similarly, for a household observed in 1985, 1986 and 1988, Group Time would create (1,2,3) while the actual values should be (2,3,5). The additional function,

```
CREATE      ; name = Map (index) $
```

can be used to handle this case. For our example, **CREATE ; Period = Map (year) $** would create the variable described above. Note that this is not specifically a panel data operation. It operates on the whole sample – the function inspects the entire set of values and determines what values should be attached to the index values. But, it does what we need for a panel. The function creates a sequence that begins at one. If you need to begin the sequence at some other value, the function can be changed to

```
CREATE      ; name = Map (index, first value) $
```

For example,

```
CREATE      ; Period = Map (year, 10) $
```

NOTE: The group count variable is used in all panel data estimators in *LIMDEP* to specify the panel. In earlier versions, a regression command,

```
REGRESS     ; Lhs = one ; Rhs = one ; Str = personid ; Panel $
```

would be used to create the variable *_groupti*, which would be identical to *ti* above. The regression form can still be used, but the preceding is likely to be simpler. In addition, in this version of *LIMDEP*, you can set the panel dimensions globally with a single ‘**SETPANEL**’ command, and the program will create the group count variables at the time they are needed. Panel data operations are discussed in Chapter R5.

The variable *time* shown above can be created internally with two functions

Group Indx (id, Pds = variable)	= sequence number from 1 to T_i within a panel,
Prd (id variable)	= sequence number from 1 to T_i within a panel.

Seq and Group Nmbr produce the same result, but they are based on different input variables. Referring to the example above, *time* could be created using Group Nmbr (*ti*) or using Seq (*personid*).

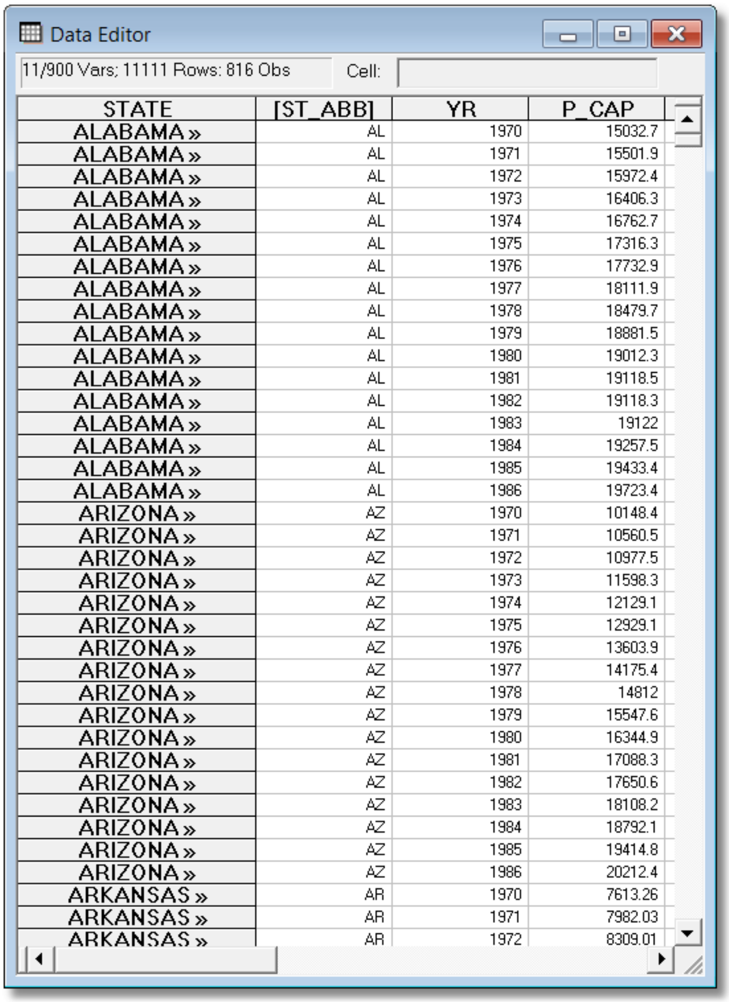
The first of these could be used to compute your own fixed effects estimator by transforming data to deviations from group means. Thus, for example, the following would produce identical results:

```
        REGRESS     ; Lhs = y ; Rhs = x ; Pds = 4 ; Panel ; Fixed Effects $
and     CREATE      ; dy = Group Devs (y, Pds = 4) $
        CREATE      ; dx = Group Devs (x, Pds = 4) $
        REGRESS     ; Lhs = dy ; Rhs = dx $
```

(The standard error produced by the second regression will be smaller because it does not correct for the degrees of freedom lost in computing the constants while the first one does.)

R4.3.18 Functions Based on Groups Using Observation Tags

Section R3.2.2 showed how to set up a set of observation tags. In the example given, the data are a panel of the lower 48 U.S. states, with 17 years of data on each. The tag variable is ST_ABB as shown in Figure R4.2.


Data Editor

11/900 Vars; 11111 Rows: 816 Obs Cell:

STATE	[ST_ABB]	YR	P_CAP
ALABAMA »	AL	1970	15032.7
ALABAMA »	AL	1971	15501.9
ALABAMA »	AL	1972	15972.4
ALABAMA »	AL	1973	16406.3
ALABAMA »	AL	1974	16762.7
ALABAMA »	AL	1975	17316.3
ALABAMA »	AL	1976	17732.9
ALABAMA »	AL	1977	18111.9
ALABAMA »	AL	1978	18479.7
ALABAMA »	AL	1979	18881.5
ALABAMA »	AL	1980	19012.3
ALABAMA »	AL	1981	19118.5
ALABAMA »	AL	1982	19118.3
ALABAMA »	AL	1983	19122
ALABAMA »	AL	1984	19257.5
ALABAMA »	AL	1985	19433.4
ALABAMA »	AL	1986	19723.4
ARIZONA »	AZ	1970	10148.4
ARIZONA »	AZ	1971	10560.5
ARIZONA »	AZ	1972	10977.5
ARIZONA »	AZ	1973	11598.3
ARIZONA »	AZ	1974	12129.1
ARIZONA »	AZ	1975	12929.1
ARIZONA »	AZ	1976	13603.9
ARIZONA »	AZ	1977	14175.4
ARIZONA »	AZ	1978	14812
ARIZONA »	AZ	1979	15547.6
ARIZONA »	AZ	1980	16344.9
ARIZONA »	AZ	1981	17088.3
ARIZONA »	AZ	1982	17650.6
ARIZONA »	AZ	1983	18108.2
ARIZONA »	AZ	1984	18792.1
ARIZONA »	AZ	1985	19414.8
ARIZONA »	AZ	1986	20212.4
ARKANSAS »	AR	1970	7613.26
ARKANSAS »	AR	1971	7982.03
ARKANSAS »	AR	1972	8309.01

Figure R4.2 Data Editor with Tag Variable

You may have a second tag variable active at any time. It may be convenient to have the second one create subgroups from the first one. **CREATE** provides a function for doing so:

```
CREATE      ; [new tagname] = Groups ([old tagname]):
              tagvalue1 (list of tags) /
              tagvalue2 (list of tags) / … $
```

The new tag variable created is alphabetic as well. Lists of tags may not overlap – a tag can only be assigned to one group. Any tags that are not assigned produce missing values for the new variable. For example, in Greene (2012), the Munnell data of this example are collected into 'regions' of states. The command to do that computation would be

```
CREATE      ; [REGION] = Groups ([ST_ABB]):
              GF (al,fl,la,ms) /                ? Gulf Coast
              MW (il,in,ky,mi,mn,oh,wi) /       ? Midwest
              MA (de,md,nj,ny,pa,va) /          ? Mid Atlantic
              MT (co,id,mt,nd,sd,wy) /          ? Mountain
              NE (ct,me,ma,nh,ri,vt) /          ? Northeast
              SO (ga,nc,sc,tn,wv,ar) /          ? South
              SW (az,nv,nm,tx,ut) /             ? Southwest
              CN (ia,ks,mo,ne,ok) /             ? Central
              WC (ca,or,wa) $                   ? West Coast
```

Figure R4.3 shows the result of the computation. Such grouping variables may be generally of use. There are four possible ways to create them. The one shown above uses a tag variable to create a tag variable. The resulting variable is a character. You can also create a tag variable based on the numeric values taken by a variable. In this case, the 'list of tags' in the preceding would be replaced by a list of values. For example,

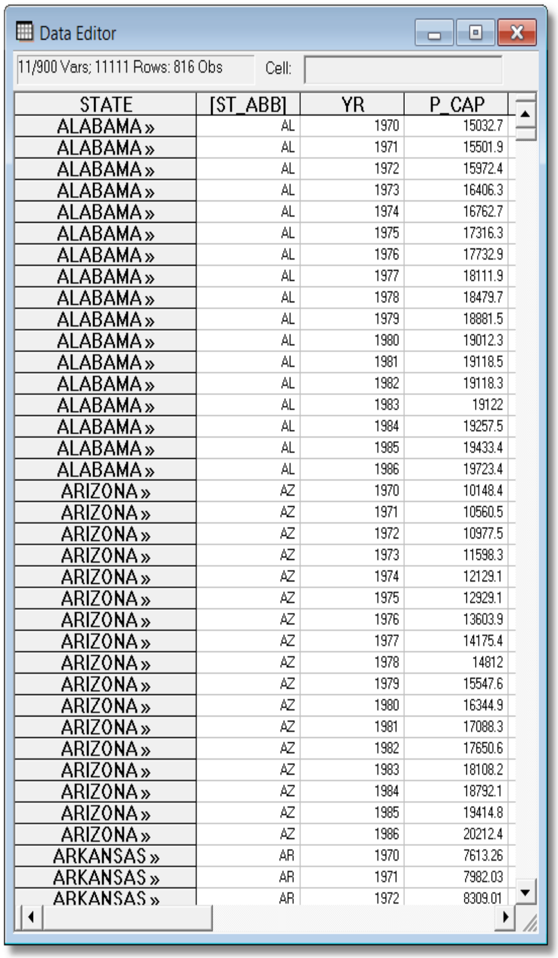
```
CREATE      ; [REGION] = Groups (yr):
              Early (1971,1972,1973) /
              Middle (1974,1975,1976) \
              Late (1977,1978,1979,1980) $
```

The Groups (…) function may be used to transform a tag to a variable or a variable to a variable as well. The general form is the same as above, with appropriate replacements of integer values and alphabetic labels. To create a numeric grouping variable from another numeric variable, for example, we might use

```
CREATE      ; ny = Groups (yr):
              1 (1970,1971,1972) /
              2 (1973,1974,1975) … $
```

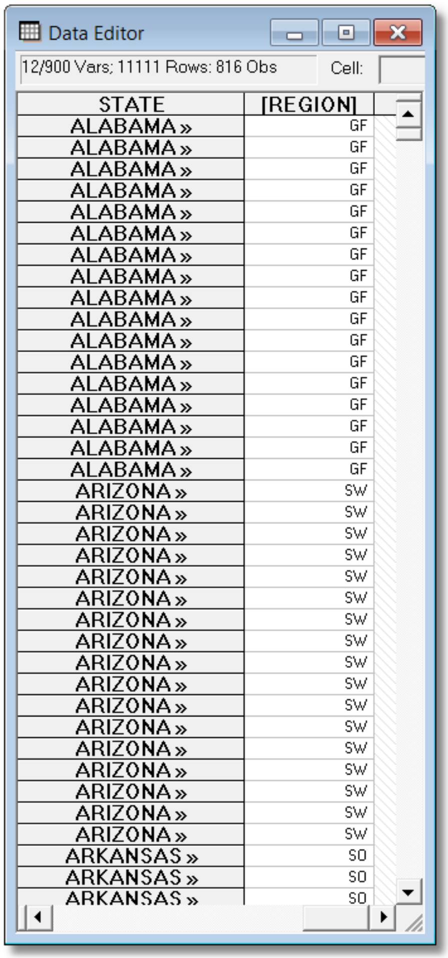
To map an alphabetic tag variable to an integer valued numeric variable, we could use, for example,

```
CREATE      ; rgn = Groups ([ST_ABB]):
              1(al,az) /
              2(co,ct) / … $
```


Data Editor

11/900 Vars; 11111 Rows: 816 Obs Cell:

STATE	[ST_ABB]	YR	P_CAP
ALABAMA»	AL	1970	15032.7
ALABAMA»	AL	1971	15501.9
ALABAMA»	AL	1972	15972.4
ALABAMA»	AL	1973	16406.3
ALABAMA»	AL	1974	16762.7
ALABAMA»	AL	1975	17316.3
ALABAMA»	AL	1976	17732.9
ALABAMA»	AL	1977	18111.9
ALABAMA»	AL	1978	18479.7
ALABAMA»	AL	1979	18881.5
ALABAMA»	AL	1980	19012.3
ALABAMA»	AL	1981	19118.5
ALABAMA»	AL	1982	19118.3
ALABAMA»	AL	1983	19122
ALABAMA»	AL	1984	19257.5
ALABAMA»	AL	1985	19433.4
ALABAMA»	AL	1986	19723.4
ARIZONA»	AZ	1970	10148.4
ARIZONA»	AZ	1971	10560.5
ARIZONA»	AZ	1972	10977.5
ARIZONA»	AZ	1973	11598.3
ARIZONA»	AZ	1974	12129.1
ARIZONA»	AZ	1975	12929.1
ARIZONA»	AZ	1976	13603.9
ARIZONA»	AZ	1977	14175.4
ARIZONA»	AZ	1978	14812
ARIZONA»	AZ	1979	15547.6
ARIZONA»	AZ	1980	16344.9
ARIZONA»	AZ	1981	17088.3
ARIZONA»	AZ	1982	17650.6
ARIZONA»	AZ	1983	18108.2
ARIZONA»	AZ	1984	18792.1
ARIZONA»	AZ	1985	19414.8
ARIZONA»	AZ	1986	20212.4
ARKANSAS»	AR	1970	7613.26
ARKANSAS»	AR	1971	7982.03
ARKANSAS»	AR	1972	8309.01


Data Editor

12/900 Vars; 11111 Rows: 816 Obs Cell:

STATE	[REGION]
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ALABAMA»	GF
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARIZONA»	SW
ARKANSAS»	SO
ARKANSAS»	SO
ARKANSAS»	SO

Figure R4.3 Observation Tags in Data Set

The program will produce in the output window a table that verifies the groups created. For example, the first setup above results in the following display:

```
------------------------------------------------------------------------
Constructed Groups
------------------------------------------------------------------------
Constructed from Tags    Stored as Tags
            [ST_ABB]     [REGION]
------------------------------------------------------------------------
Group ID   Components
------------------------------------------------------------------------
GF         AL        FL        LA        MS
MW         IL        IN        KY        MI        MN        OH
           IL        IN        KY        MI        MN        OH
MA         DE        MD        NJ        NY        PA        VA
MT         CO        ID        MT        ND        SD        WY
NE         CT        ME        MA        NH        RI        VT
SO         GA        NC        SC        TN        WV        AR
SW         AZ        NV        NM        TX        UT
CN         IA        KS        MO        NE        OK
WC         CA        OR        WA
------------------------------------------------------------------------
```

R4.3.19 Aggregating Grouped or Stratified Data

The panel data functions listed in Section R4.3.17 are useful for aggregating the within group data. Functions are provided for similar operations for grouped data such as in this example. Referring to Figure R4.3, which shows the 17 observations (1970-1986) for Alabama then Arizona and so on, in order to compute the state means of the variable *p_cap*, we could use the usual panel data operations,

```
CREATE      ; stnum = Trn(17,0) $
SETPANEL    ; Group = stnum ; Pds = ti $
CREATE      ; pcapbar = Group Mean (p_cap, Pds = ti) $
```

Consider a different aggregate. According to the Groups function, the Gulf Region consists of AL, FL, LA and MS which are the 1st, 8th, 16th and 22nd states in the data set. There are 17 years of data on each state. We need the year means for the four states. The following does this calculation for the regions:

```
CREATE      ; [By ([REGION], yr)] pcapbar = Xbr(p_cap) $
```

By this instruction, the variable *pcapbar* contains for each distinct year within a group (region) the mean of the observations for the elements of that group. For this example, for 1971, in the Gulf region, each of the four states will have the same value, the 1971 mean of those four states. The computation is computed for each distinct year, for each region. The stratification indicator in the By (…) specification may also be a variable. For an example, suppose the strata are school districts in counties within a state and the observations are SAT test scores. The following would obtain the mean test scores by year for the observations within the districts:

```
CREATE      ; [By (District, year)] meantest = Xbr(sat) $
```

The stratification may also be omitted. To obtain means by an index variable when the data are not arranged in the usual panel format, you may use

```
CREATE      ; [By (year)] variable = Xbr(variable) $
```

For example, for the data in Figure R4.3, the command

```
CREATE      ; [By (yr)] pcapbar = Xbr(p_cap) $
```

creates the year specific means. In each year, each of the 48 states receives the mean of the 48 observations on *p_cap* for that year.

The general format of the command is,

```
        CREATE ;    [By ([tag variable], within group index)] variable = Xbr(variable)
or
        CREATE ;    [By (variable, within group index)] variable = Xbr(variable)
or      CREATE ;    [By (within group index)] variable = Xbr(variable)
```

The function Xbr requests the mean of the group members. Other functions that may be used are Logxbr for the log of the mean (if possible), Sum and Logsum. Multiple aggregations may be specified by separating them by slashes. For example, to do all of the aggregating needed for the example in Greene (2012), the instruction could be

```
CREATE      ; [By ([REGION],yr)]
              lgsp     = Logsum(gsp) /
              lpc      = Logsum(pc) /
              lhwy     = Logsum(hwy) /
              lwater   = Logsum(water) /
              lutil    = Logsum(util) /
              lemp     = Logsum(emp) /
              ur       = Xbr(unemp,emp) $
```

R4.4 Random Number Generators

Many operations that you and *LIMDEP* do involve random number generation. This includes bootstrapping, mixed model estimation, model simulation, and any number of types of experimental operations that you will perform with the program. At the heart of all of these calculations is the random number generator (RNG) – every modern computer program contains one. *LIMDEP* has two, one by L'Ecuyer (1999) and a second named the Mersenne Twister, both discussed in Appendix R4A.3. Both RNGs have excellent properties (such as periods up to 2^{10000}). The Mersenne Twister has recently been built into other mathematical programs such as *MATLAB*. We do not have a preference for either of these; the Mersenne Twister is the default. You can switch between the two by using the **CALC** command

```
CALC        ; Rng(1) $      to set the RNG to be L'Ecuyer's
CALC        ; Rng(2) $      to set the RNG to be the Mersenne Twister.
```

Once the generator is set, all subsequent draws for all purposes are produced by the chosen generator. (It would not be natural to do so, but you can switch back and forth between these two RNGs at will. The properties of a sequence of values are not affected by which generator you use, or even if some draws are taken with one RNG and the rest with the other.)

The central function of a (pseudo) RNG is to create series of values that appear to be random strings of draws from the standard uniform distribution. Random draws from other distributions are obtained by transforming the $U[0,1]$ values. To draw a sample from a continuous uniform distribution in the indicated range,

```
CREATE      ; name = Rnu(0,1) $
```

As noted, this is the 'primitive' operation of random number generation. *LIMDEP* provides roughly 20 different functions for generating random samples from different distributions.

R4.4.1 Setting the Seed for the Random Number Generator

Given your choice of RNG, a second consideration for you as user is the seed of the RNG. Random number generators generate strings of pseudorandom numbers – they are not really random, and it is possible to generate the same string twice (which establishes the nonrandomness of the string). But, the string of values generated will look enough like a set of random numbers that they can reliably be used for the calculations for which we need them for. An RNG produces a deterministic string of NP values, then the NP+1 value is equal to the first, and it starts over and repeats. NP is the period of the generator. Early primitive generators, such as IBM's scientific subroutine package, had a period of 2^{31}-1, which relates to the use of a 32 bit word inside the digital computer. This period is unsatisfactory for modern research. The L'Ecuyer generator has a period of about 2^{132}. The Mersenne Twister has a period about 2^{10000} – for practical purposes, it never repeats. The seed of an RNG is a pointer to where in its period of values the cycle begins. The usefulness of this feature is that if one can set the seed, they can reproduce the string of values (regardless of how long that string is). For you to replicate any results using random numbers, you need to be able to set the seed of the RNG. To set the seed for the random number generator, use the command

```
CALC          ; Ran(seed) $
```

In this fashion, you can replicate a sample from one session to the next. Use a large (e.g., seven digit) odd number for the seed. The value does not matter as long as it is a large number.

An RNG works by using the seed to compute the pseudorandom value, then it resets the seed (pseudorandomly) for the next value. You can find out what the current seed is with

```
CALC          ; Peek ; Ran(0) $
```

We're not sure this is useful, since it only tells you where you are going, nothing about where you have been. You can't use this value to reproduce any calculations already done – it's too late. But, we expect that users will find uses that we have not thought of, so the preceding is made available.

R4.4.2 Basic Random Number Generation

After generation of primitive U[0,1] draws, there are two essential functions, general uniform and general normal random variable sampling. The first of these generates values distributed uniformly from *lower* to *upper* by

$$U[lower,upper] = lower + (upper - lower) \times U[0,1].$$

This is done internally. The command for this is

```
CREATE        ; name = Rnu(lower limit, upper limit) $
```

The second essential function is random draws from the standard normal distribution. The function is

```
CREATE        ; name = Rnn(0,1) $
```

HOW IT'S DONE: Draws from the normal distribution are generated many ways, using draws from U[0,1]. A common method also used for other distributions is the inverse probability transformation; $x(\text{normal}[0,1]) = \Phi^{-1}(U[0,1])$ where Φ^{-1} is the inverse of the standard normal CDF. *LIMDEP* uses a transformation of a pair of random draws developed by Box, Muller and Marsaglia. See the appendix to this chapter for details on this computation.

The next essential step is producing values from the general normal population, $N[\mu,\sigma^2]$. This is obtained by

$$N[\mu,\sigma^2] = \mu + \sigma N[0,1].$$

The command syntax for this normal simulation is

```
CREATE        ; name = Rnn(mean, standard deviation) $
```

which will create a variable containing a sample from the indicated normal distribution.

The next section details the 20+ different distributions from which random samples may be drawn. The general command is

```
CREATE        ; name = Rng(parameters) $
```

where **Rng** is the three letter symbol for the distribution and **parameters** are values such as (μ,σ) needed to do the simulation. The sample is placed with the observations in the current sample. If you want to draw more than the default number, you might want to use the **ROWS** command (see Section R3.4) before you draw the sample.

Random draws may also appear anywhere in an expression as operands whose values are random draws from the specified distribution. For example, a random sample from a chi squared distribution with one degree of freedom could be drawn with

```
CREATE        ; name = Rnn(0,1) ^ 2 $
```

(There is an easier way, though.) Random samples can be made part of any other transformation. For example, the following shows how to create a random sample from a regression model in which the assumptions of the classical model are met exactly:

```
CREATE        ; x1 = Rnu(10,10)
              ; x2 = Rnn(16,10)
              ; y   = 100 + 1.5 * x1 + 3.1 * x2 + Rnn(0,50) $
```

The regression of y on $x1$ and $x2$ would produce estimates of $\beta_1 = 100$, $\beta_2 = 1.5$, and $\beta_3 = 3.1$ and a residual standard deviation, s_e, close to 50.

In addition to the Rnn(m,s) (normal with mean m and standard deviation s) and Rnu(l,u) (continuous uniform between l and u), you can generate random samples from continuous, discrete and multivariate normal distributions. There are described in the following sections.

R4.4.3 Random Samples from Continuous Distributions

Rng(*m*,*s*)	= lognormal with parameters *m* and *s*
Rnt(*n*)	= *t* with *n* degrees of freedom
Rnx(*d*)	= chi squared with *d* degrees of freedom
Rnf(*n*,*d*)	= *F* with *n* numerator and *d* denominator degrees of freedom
Rne(*q*)	= exponential with mean *q*
Rnw(*a*,*c*)	= Weibull with location a and scale *c*. If *c* = 1, use Rnw(*a*)
Rnh(*a*,*c*)	= Gumbel (extreme value) with location *a*, scale *c*. If *c* = 1, use Rnh(*a*)
Rni(*a*,*c*)	= gamma with scale a and shape c. If *a* = 1, use Rni(*c*)
Rna(*a*,*b*)	= beta with parameters *a* and *b*
Rnl(0)	= logistic
Rnc(0)	= Cauchy
Rno(0)	= symmetric triangular [-1,+1], (1) = [0,2], (-1) = [-2,0], (c,x) = [-x×c,+x×c]
Rns(0)	= inverse Gauss, Rns(μ,σ)
Rnz(s1,s2)	= skew normal, s1×z1 + s2*|z2|
Rnf(k1,k2)	= F distribution with k1 and k2 degrees of freedom.

For sampling from the noncentral chi squared population, use the function is Rnx(*d*,*a*)**,** where *d* is the degrees of freedom and *a* is the noncentrality parameter. This could be done with

Rnx(d-1) + Rnn(a,1)^2,

so this automates the noncentrality parameter. Sampling from the singly (numerator only) noncentral F is done with

(Rnx(n,a)/n)/(Rnx(d)/d).

A doubly noncentral F variable can also be created by having noncentral chi squared variables in both numerator and denominator. One would not normally use this in econometric work, however.

Halton sequences are often used as alternatives to strings of pseudo-random values. Halton sequences are not random, but when used in computing integrals, they provide the same (but better) coverage of the desired range of variation, as if they were random draws. Halton sequences are deterministic functions of (*i*,*h*) where i is in the sequence of integers (*i* = 1,…,) and *h* is a prime number. The sequence of values lies in the unit interval. The string is then transformed to the desired type of draws using the inverse of the CDF. For example, to generate a string of values for the standard normal distribution, one would use

$$z_i = \Phi^{-1}[H(i,h)].$$

To generate a sequence of Halton draws for your sample, use

```
CREATE      ; x  =  Hlt(h) $
```

where *h* is a prime number less than or equal to 97. For example, a sequence might be generated using

```
CREATE      ; x = Inp(Hlt(7)) $.
```

It is often desirable to start the sequence at a value larger than one to avoid the possible correlation among the first few values. You can specify a burn in period by adding the value to the Hlt function. For example,

```
CREATE      ; x = Inp(Hlt(7,10)) $
```

would begin the preceding sequence at i = 11 (i.e., discarding the first 10 values).

R4.4.4 Random Samples from Discrete Distributions

Rnp(q)	= Poisson with mean q
Rnd(n)	= discrete uniform, x = 1,...,n
Rnb(n,p)	= binomial, n trials, probability p
Rnm(p)	= geometric with success probability p
Rnj(q,λ,P)	= negative binomial – P. Use Rnj(q,λ) if P = 2 (standard)

For sampling from the binomial distribution, The limits on n and p are $n$log(p), and $n$log(1-p) must both be greater than -264 to avoid numerical overflow errors.

HOW IT'S DONE: See the appendix to this chapter for details on this computation.

You must provide the 'a' in the Weibull and Gumbel and the '0', logistic, and Cauchy functions. You may also sample from the truncated standard normal distribution. Two formats are

Rnr(*lower*)	= sample from the distribution truncated to the left at '*lower*'
Rnr(*lower*,*upper*)	= distribution with both tails truncated

HOW IT'S DONE: See the appendix to this chapter for details on this computation.

E.g., Rnr(.5) samples observations greater than or equal to .5

Parameters of all requests for random numbers are checked for validity. For the truncated normal, you must have

$$lower \leq 1.5, upper \geq -1.5, \; upper - lower \geq .5$$

If 'upper' is not provided, it is taken as +∞. If you need upper truncation, a transformation which will produce the desired result is -Rnr(-*lower*).

The parameters of any random number generator can be variables, other functions, or expressions, as well. For example, you might simulate draws from a Poisson regression model with

```
CREATE      ; x1 = Rnn(0,1)
            ; x2 = Rnu(0,1)
            ; y  = Rnp(Exp (.2 + .3 * x1 - .05 * x2 )) $
```

R4.4.5 Sampling from the Multivariate Normal Distribution

To sample from the multivariate normal distribution, it is necessary to generate a set of random variables. We do this by using the following theoretical result.

> If $\mathbf{x} = (x_1,...,x_K)$ are distributed with joint normal distribution with mean vector **0** and covariance matrix **I**, then $\mathbf{Ax} + \boldsymbol{\mu}$ is distributed multivariate normally with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{AA}'$.

You can use this result to generate a multivariate sample from the normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ by simply decomposing $\boldsymbol{\Sigma}$ into $\mathbf{AA}'$, and using this and the desired $\boldsymbol{\mu}$ in the theoretical result. We use the Cholesky decomposition in which **A** is a lower triangular matrix. The operation will create a multivariate sample – that is *K* variables where *K* is the number of elements in *x* and *N* observations, where *N* is the number of observations in the current sample. You can sample from the distribution with up to 100 elements, in which case, you will create 100 new variables in your data area. Collectively, these *K* variables are a multivariate sample from the specified multivariate normal distribution.

The command for generating a sample from the multivariate normal distribution is

```
CREATE      ; name = Rmn(vector μ,  matrix Σ) $
```

You must provide the **vector μ** and **matrix Σ**. However, if you want **μ** to equal zero, omit it. Thus,

```
CREATE      ; name = Rmn(matrix Σ) $
```

samples from the multivariate normal population with mean vector zero and covariance matrix **Σ**. Alternatively, you can force **Σ** to be an identity matrix by using

```
CREATE      ; name = Rmn(vector μ) $
```

to sample from the multivariate normal population with mean vector **μ** and covariance matrix **I**. Finally, if you want to sample from the standard normal population with mean vector zero and covariance matrix **I**, use

```
CREATE      ; name = Rmn(K) $
```

where *K* is the number of elements in the random vector. In this case, *K* must either be an integer from 1 to 100 or the name of scalar which contains an integer from 1 to 100. *LIMDEP* detects what kind of sample you want to by examining what appears in the parentheses. A vector and a matrix implies the first case, just a matrix implies the second, just a vector implies the third, and just a number implies the fourth.

The '**; name =** ' specifies the name of a namelist that will be created. This may be a new namelist or you can replace an existing one. The variables in that namelist will be constructed as if the command were

```
NAMELIST    ; name = name00,name01,... $
```

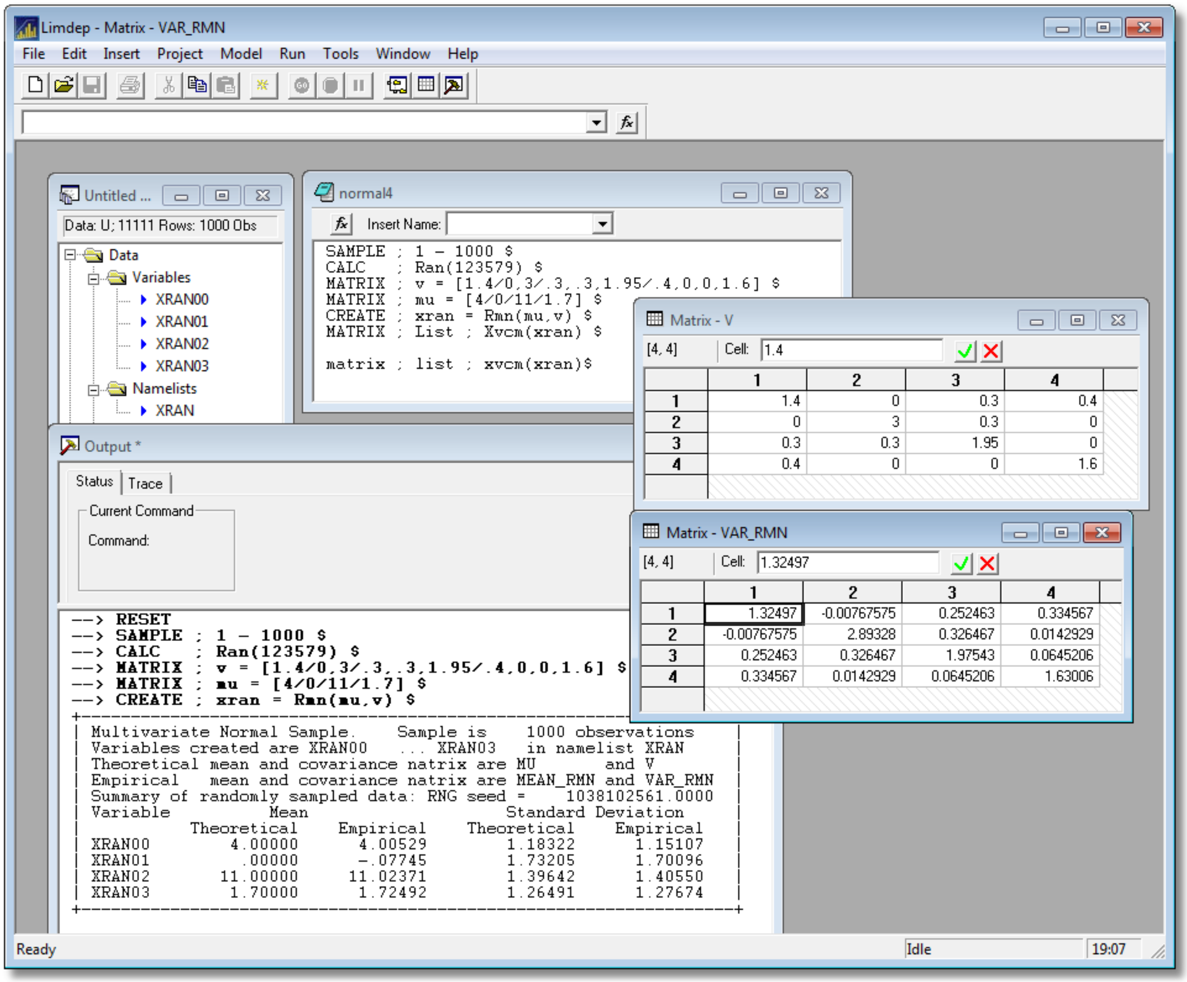
For example, if you use

```
CREATE      ; xran = Rmn(mu,v) $
```

where *mu* is a 10×1 vector and *v* is a 10×10 covariance matrix, then there will be a new namelist created in your data area:

xret = xran00,xran01,xran02,...,xran09.

This routine creates the variables, and issues a report of what it has computed. The following shows an example of sampling 1,000 observations from a 4-variate normal distribution.



```
SAMPLE ; 1 - 1000 $
CALC   ; Ran(123579) $
MATRIX ; v = [1.4/0,3/.3,.3,1.95/.4,0,0,1.6] $
MATRIX ; mu = [4/0/11/1.7] $
CREATE ; xran = Rmn(mu,v) $
MATRIX ; List ; Xvcm(xran) $

matrix ; list ; xvcm(xran)$
```

Matrix - V [4, 4] Cell: 1.4

	1	2	3	4
1	1.4	0	0.3	0.4
2	0	3	0.3	0
3	0.3	0.3	1.95	0
4	0.4	0	0	1.6

Matrix - VAR_RMN [4, 4] Cell: 1.32497

	1	2	3	4
1	1.32497	-0.00767575	0.252463	0.334567
2	-0.00767575	2.89328	0.326467	0.0142929
3	0.252463	0.326467	1.97543	0.0645206
4	0.334567	0.0142929	0.0645206	1.63006

```
--> RESET
--> SAMPLE ; 1 - 1000 $
--> CALC   ; Ran(123579) $
--> MATRIX ; v = [1.4/0,3/.3,.3,1.95/.4,0,0,1.6] $
--> MATRIX ; mu = [4/0/11/1.7] $
--> CREATE ; xran = Rmn(mu,v) $
+------------------------------------------------------------------+
| Multivariate Normal Sample.    Sample is    1000 observations    |
| Variables created are XRAN00   ... XRAN03   in namelist XRAN     |
| Theoretical mean and covariance natrix are MU         and V      |
| Empirical   mean and covariance natrix are MEAN_RMN and VAR_RMN  |
| Summary of randomly sampled data: RNG seed =   1038102561.0000   |
| Variable            Mean                  Standard Deviation     |
|            Theoretical    Empirical    Theoretical    Empirical  |
| XRAN00          4.00000      4.00529        1.18322      1.15107 |
| XRAN01           .00000     -.07745         1.73205      1.70096 |
| XRAN02         11.00000     11.02371        1.39642      1.40550 |
| XRAN03          1.70000      1.72492        1.26491      1.27674 |
+------------------------------------------------------------------+
```

Figure R4.4 Sampling from the Multivariate Normal Population

Note in the report in the output window, the theoretical and empirical means and variances are both reported. The actual mean and standard deviations of the drawn sample will not equal the theoretical ones, since the data are a random sample – they are not constrained. Also, the report shows the seed for the random number generator. It does not equal the seed that appears in the command in the editing window. The **CALC ; Ran(seed) $** function allows you to set a specific seed for the random number generator. The actual value used internally is a transformation of the one you give. The point of the function is to enable you to reset the seed to the same value, not a particular value. Specific values of the seed are meaningless. But, your ability to reset the seed to a specific value allows you to replicate random sampling results.

This procedure creates several results:

- The namelist as specified in the command.
- The variables (up to 100 of them) which are the random sample.
- Matrices *mean_rmn* which is the matrix of means of your sample, and *var_rmn* which is the sample covariance matrix.

The latter two matrices could be created immediately after the sampling command with

```
MATRIX      ; mean_rmn = Mean(namelist)
            ; var_rmn = Xvcm(namelist) $
```

All of the elements of the setup for this computation are checked internally before any computation is done. The following conditions will generate diagnostics:

- Your matrices *mu* and *v* are not currently in the matrix names table.
- Your parentheses contain more than two names.
- The matrix is not square.
- The vector is not conformable with the matrix. *Mu* may be a row or a column, but it must be the same size as *v*, whichever applies.
- Your computation implies more than 100 variables.
- You are out of space for new namelists or variables.
- Your matrix *v* is not symmetric.
- Your matrix *v* is not nonnegative definite.

If none of these failures occur, the computation will proceed. For the last of these conditions, *LIMDEP* checks the characteristic roots of your matrix. If none are negative, we proceed. (A zero root, indicating singularity is OK. If your matrix were [1,1/1,1], this is singular but it is nonnegative definite.

R4.5 Compound Names for Variables

The names of variables and scalars may be of the form *aaaa:ssss* where *ssss* is the name of a scalar. The scalar must take an integer value from 00 to 99. The value is appended to the name to make a variable with the compound name. This feature will be useful for looping in procedures. For example:

```
CALC        ; index = 1 $
PROC $
CREATE      ; x : index = 1 / index $
ENDPROC $
EXECUTE     ; index = 1,10 $
```

creates 10 variables, *x*1 = 1, *x*2 = 1/2, *x*3 = 1/3, *x*4 = 1/4, ..., *x*10 = 1/10.. The Brant test for homogeneity in an ordered logit model provides another example – in this program, both matrices and variables are being given compound names. The commands below show only the part of the program that uses the feature described here. The full procedure with the remaining analysis and additional comments appears in Chapter R16.

```
?==========================================================================+
? This is an analysis of an ordered choice variable y=0,1,...maxy.
? Here, we are generating artificial data.
?==========================================================================+
 SAMPLE   ; 1-1000$
 CALC     ; Ran (12345) $
 CREATE   ; y = Rnd(6)-1 ; xa = Rnn(0,1); xb = Rnn(0,1); xc = Rnn(0,1)$
 NAMELIST ; x = xa,xb,xc $  x does not include a constant term.
 NAMELIST ; x1 = x,one $
 CALC     ; k = Col(x) $
 CALC     ; ymax = max(y) ; y1 = ymax-1 ; kj = ymax* k ; k1j = y1*k $
 MATRIX   ; i = Iden(k) ; z = Init(k,k,0) ; mi =-1*i $
 MATRIX   ; bt = Init(kj,1,0.) ; d = Init(k1j,kj,0.) $
?==========================================================================+
? This procedure computes the individual logit equations. To reduce the   |
? number of commands, it makes heavy use of compound names.               |
? Loop index y1 takes values 1,2,...,ymax.  j = y1 – 1 = 0,1,2,...ymax-1  |
? The procedure is creating variables z0, z1, ... each equal to a binary  |
? variable that equals 1 when y > j.  It is creating coefficient vectors  |
? b0, b1, ... then injecting (stacking) them in the large vector bt.      |
? Each LOGIT command creates a variable with fitted probabilities, p0,... |
? After each b:j is computed, a vector of derivatives, w0=p0(1-p0), w1=   |
? p1(1-p1),... is computed. We are creating matrices v0, v1,... as        |
? inverses of moment matrices. Finally, the large matrix D is a           |
? partitioned matrix in which block row j contains I on the diagonal and  |
? –I at the end of the row.                                               |
?==========================================================================+
 PROC = LOGITS$
    CALC     ; j = y1 - 1 ; jy = j*k+1 ; jyk = jy+k $
? This command creates variables z0, z1, ...
    CREATE   ; z:j = y > j $
? The probabilities kept by this procedure are p0, p1, ...
    LOGIT    ; Lhs = z:j ; Rhs = x1 ; Prob = p:j $
? The capability is also available in MATRIX. This creates b0, b1, ...
    MATRIX   ; b:j = b(1:k) ; bt(jy) = b:j $
? The previously created p0, p1,... are used to create w0, w1, ...
    CREATE   ; w:j = p:j*(1-p:j) $
    CALC     ; jy = min(jy,((ymax-2)*k+1)) ; jyk = jy+k $
? This MATRIX command computes v0, v1, ...
    MATRIX   ; v:j = <x1'[w:j]x1> ; vt=v:j ; vt = part(vt,1,k,1,k); v:j = vt $
    MATRIX   ; d(jy,1) = I ; d(jy,jyk) = mi$
 ENDPROC $
 EXECUTE     ; y1 = 1,ymax ; Silent $
```

R4.6 Changing Particular Observations of a Variable

Once a column of data has been entered, there are several ways to edit it, if necessary. The data editor discussed in Chapter R3 can be used directly to change values of a variable. Note, though, that the data editor can only access the first 5,000 rows of the data area. You can also use the command

```
CREATE      ; variable (observation) = new value ; ... $
```

to replace any specific observations. Up to 50 replacements may appear in a single **CREATE** command. If the data are time series data, specified with the **DATES** command, the observation number will be a date, instead. For example,

```
CREATE      ; gnp (1976.1) = 2105.729 $
```

R4.7 Recoding Variables – The RECODE Command

The **RECODE** command allows you to change the values taken by one or more variables to a set of other values. This can replace up to 50 **If (...)** then, **(Else)** ... sorts of **CREATE** commands with a single instruction. The syntax of the **RECODE** command is

```
RECODE      ; variable(s) to recode  (same recoding is applied to each)
            ; old values = new value
            ; old values = new value
            ; ... (up to 50 of these) ...
            ; * = default value $
```

The command shown replaces the original variable with the transformed one. To recode a variable into a new one, use

```
RECODE      ; old variable -> new variable ; … $
```

‘Old values’ are as many as 20 particular values, such as 1,2,3,4,5 = 77.77. This would transform all occurrences of any of the five values on the left to 77.77. Ranges of values may be specified as

```
            lower / upper = new value
```

which transforms any value found in the range lower to upper, inclusive, to the new value. For example, -5.234 / 1.297 = 9transforms any value from -5.234 to 1.297 to 9. The last specification (***** = default) is optional and specifies the default value to be used if the value found for that observation is not in any of the recode specifications. If no default is given, the original value is left intact. For example,

```
RECODE      ; a ; 1 / 10 = 10 $
```

changes any observation on *a* from 1 through 10 to 10. All other values of *a* remain unchanged.

For example, suppose *income* is given in dollar figures, with values -1, -2, and -3, indicating missing data for three different reasons. We convert these all to -999, the income ranges to a simple grouped coding, and all values not found in the given ranges to 0;

```
RECODE    ; income
          ; -1,-2,-3 = -999
          ; 0 / 15000 = 1; 15001 / 35000 = 2
          ; 35001 / 9999999 = 3 ; * = 0 $
```

Lists of values and ranges of values may not both appear in the same specification. But, since recodings need not have different values on the right, you can just give a separate specification for each. **RECODE** specifications are processed sequentially and later ones can override earlier ones. For example, 1,2,3 = 88 ; 3,4,5 = 99 transforms the value 3 to 99, not 88. The three parts of the command must be given in the order shown above. In particular, all specifications after a **; * = default** are ignored.

The original variable is lost after the recoding. If you want to keep a copy of it, precede the **RECODE** command with

```
CREATE    ; copy = variable to be recoded $
```

R4.8 Sorting Variables – The SORT Command

You can sort a variable while carrying any number of other variables. The command is

```
SORT      ; Lhs = key variable [; Rhs = variables to carry] $
```

The key variable in the Lhs is sorted in ascending order. To obtain a sort in descending order, add **; Descending** at the beginning of the command. To sort just one variable, omit the **; Rhs = list** part of the command. This command produces no output except for a simple message which indicates that the sort was completed successfully.

As with most other data manipulation commands, the **SORT** command is applied to the current sample, not the entire data set. If you wish to sort the entire data set, you can either reset the sample or just add **; All** to the **SORT** command. For example, if your current sample is 1-10, 21-40 and you give a **SORT** command followed by a **LIST** command, the listed data will be sorted. But, if you follow your **SORT** with **SAMPLE ; All $**, *then* list, observations 1-10 and 21-40 will be sorted, but others will not. To sort your entire data set keying on a variable, you should use

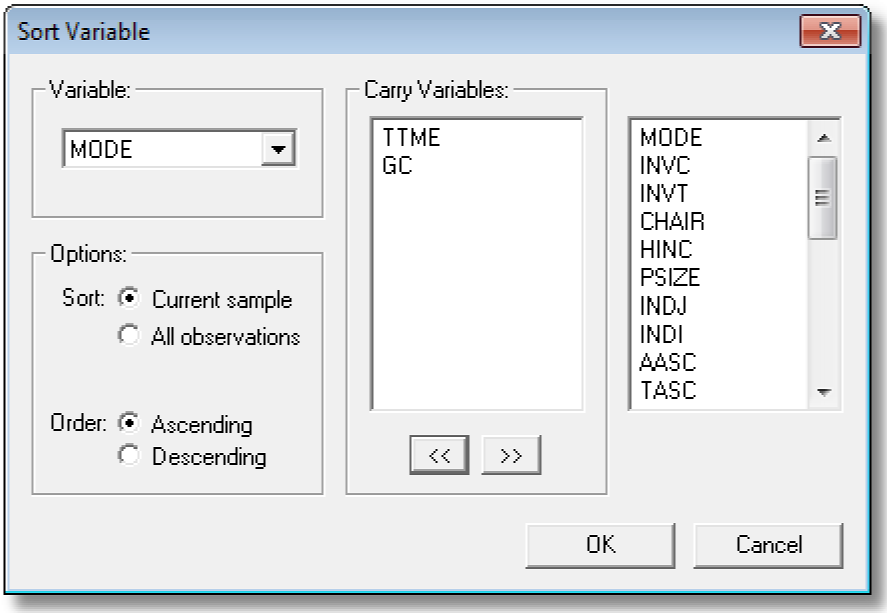
```
SAMPLE    ; All $
SORT      ; Lhs = key variable ; Rhs = * $
```

SORT may be invoked from the Project:Sort Variable menu or by selecting, then right clicking any variable name in the project window. The maximum number of observations that can be sorted is 250,000. The dialog box for **SORT** is shown in Figure R4.5.

SORT does not automatically carry the observation labels with the variable being sorted. (See Section R3.5.4.) In order to do so, add

; Labels

to the sort command. Do note, however, that unless you carry all variables with the sort key, the labels will be inconsistent with the observations, either those sorted if you do not carry the labels, or those not sorted if you do.


Sort Variable

Variable: MODE

Carry Variables: TTME, GC

MODE, INVC, INVT, CHAIR, HINC, PSIZE, INDJ, INDI, AASC, TASC

Options:
Sort: [✓ Current sample] All observations
Order: [✓ Ascending] Descending

Figure R4.5 Dialog Box for SORT

You want to keep in mind, when you sort a variable, the correspondence between it and other variables in your sample is lost. There are two ways to avoid this. One way is simply to carry the rest of the sample with the variable of interest. Use

```
SORT        ; Lhs = the interesting variable ; Rhs = * $
```

This reorders the entire data set according to this variable. Another way that may be more attractive is to carry an index variable that will allow you to undo the sort later. Consider an example:

```
FRONTIER    ; Lhs = logy ; Rhs = logx ; Eff = ui $
CREATE      ; index = Trn(1,1) $  Observation index
SORT        ; Lhs = ui ; Rhs = index ; Labels $
```

Operate on Plot, List, etc. using your *ui* variable, now sorted.

```
SORT        ; Lhs = index ; Rhs = ui ; Labels $  This undoes the sort.
```

R4.9 The DELETE and RENAME Commands

Two commands which should rarely be necessary are

DELETE ; list of variables $

and

RENAME ; old name = new name $

Use the second to change the name of a variable. The first may be useful if you have many observations and are running out of space in your data area as you create variables.

Both **DELETE** and **RENAME** can be invoked by right clicking any variable name in the project window, as shown in Figure R4.6.

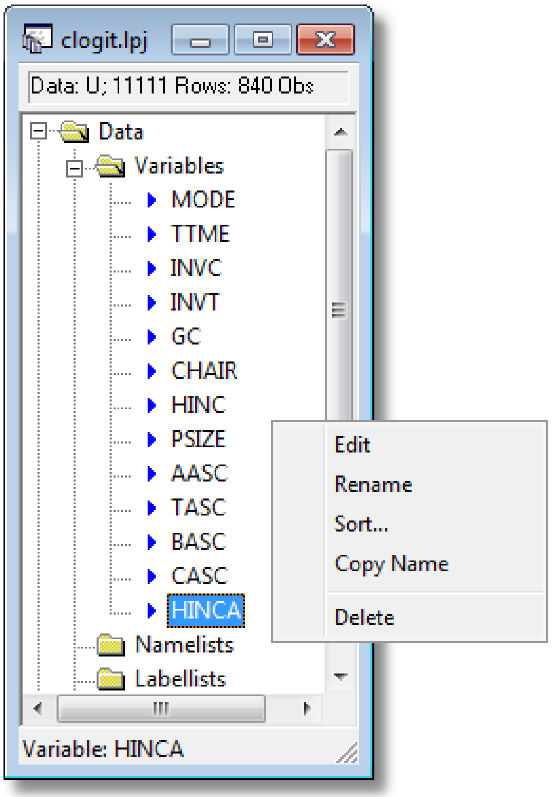


Figure R4.6 Rename and Delete Options from Project Window

If you select Rename, the variable name will be framed in a box, and can be edited or replaced, in place. To delete a variable you can select Delete in the menu or just highlight the variable (or matrix, namelist, or scalar) name in the project window, then press the Del key on your keyboard. You will be asked for confirmation.

Appendix R4A Numerical Methods

R4A.1 Computing Bivariate Normal Probabilities

Standardized (zero means, unit variances) bivariate normal probabilities B(x,y,ρ) are computed using a 15 point Gauss-Laguerre quadrature. The integration is done in one dimension by rewriting the bivariate distribution as the product of the marginal distribution of x times the conditional distribution of y given x. During the integration, we use the error function to restrict the range of integration to [0,∞), and use the in line Φ(.) function – integral of the univariate standard normal distribution as the integral within the integral. Let w_i denote the Laguerre weights and h_i denote the nodes. The formulation used is as follows:

$$d_1 = \mathbf{1}\,(x< 0),\; d_3 = 1 - 2d_1$$
$$d_2 = \mathbf{1}\,(y< 0),\; d_4 = 1 - 2d_2$$
$$V = \sum_{i=1}^{15} w_i\left[2\Phi(a_i)\frac{1}{5.0132564549262001}\exp\left(h_i - \frac{1}{2}(h_i + d_3x)^2\right)\right]$$
$$a_i \;=\; d_4\,(d_3\rho h_i + \rho x - y) \,/\, (1 - \rho^2)^{1/2}$$
$$\text{Prob}[X\geq x,\, Y\geq y] \;=\; d_3d_4V - d_1d_2 + d_1\Phi(-y) + d_2\Phi(-x).$$

Nodes and weights for the quadrature are as follows:

$$\begin{array}{llll}
h_1,\ldots,h_{15} = & 0.02110687265306352, & 0.11122304843701245, & 0.27339875290117911, \\
 & 0.50775546039766938, & 0.8144213676108329, & 1.193559990964792, \\
 & 1.645373297397144, & 2.1701027938568, & 2.7680303764366516, \\
 & 3.4394792198475525, & 4.1848147744876557, & 5.0044458955656313, \\
 & 5.8988261184898432, & 6.8684550925062301, & 7.913880184774997 6. \\
w_1,\ldots,w_{15} = & 0.05303709733976105, & 0.11284582465517608, & 0.15082452315872363, \\
 & 0.16279133631194213, & 0.15185641060466367, & 0.12593625823209979, \\
 & 0.09419839305849645 3, & 0.0640788141334108, & 0.0398456458245284, \\
 & 0.02272413644209539, & 0.01191223548930554, & 0.005748310643806657, \\
 & 0.00255593490701438, & 0.001047812282114606, & 0.000396170048170894.
\end{array}$$

R4A.2 Computing Multivariate Normal Probabilities

We use the GHK simulator for this computation. The full method is detailed in Greene (2012), so we provide only a sketch here. The desired probability is Prob[$a_i\leq x_i\leq b_i$, i = 1,...,K], where the K variables have zero means and covariance matrix **Σ**. (Nonzero means are accommodated just by transformation to simple deviations.) The probability is approximated by

$$P = \frac{1}{R}\sum_{r=1}^{R}\prod_{k=1}^{K} Q_{rk}$$

where R is the number of points used in the simulation. The Cholesky factorization of **Σ** is **LL′** where **L** = $[l]_{km}$ is lower triangular. Note $l_{km} = 0$ if $m>k$. The recursive computation of P is begun with $Q_{r1} = \Phi(b_1/l_{11}) - \Phi(a_1/l_{11})$, where Φ($t$) is the standard normal CDF evaluated at t. Using the random number generator, ε_{r1} is a random draw from the standard normal distribution truncated in the range $A_{r1} = a_1/l_{11}$ to $B_{r1} = b_1/l_{11}$. The draw from this distribution is obtained using Geweke's method. For a draw from the $N[\mu,\sigma^2]$ distribution truncated in the range A to B, we obtain u = a draw from the U[0,1] distribution. Then, the desired draw is $z = \mu + \sigma\Phi^{-1}[(1-u)\Phi((B-\mu)/\sigma) + u\Phi((A-\mu)/\sigma)]$.

For k = 2,...,K, use the iteration

$$A_{rk} = \left[a_k - \sum_{m=1}^{k-1} l_{km}\varepsilon_{rm}\right] / \, l_{kk}\,, \quad B_{rk} = \left[b_k - \sum_{m=1}^{k-1} l_{km}\varepsilon_{rm}\right] / \, l_{kk}\,,$$

$$Q_{rk} = \Phi(B_{rk}) - \Phi(A_{rk}).$$

Then, P is then the average of the R draws of products of K probabilities. Numerical properties and efficiency of this simulator are discussed at many places in the literature. References are given in Greene (2012).

You can set the number of draws globally (that is, for all uses of the simulator) with the command **CALC ; Rep(R) $** where R is the number you desire. The model specification **; Rep = R** on any model command has the same effect.

R4A.3 Uniform Random Number Generation

The core of *LIMDEP*'s (and every other program's) routines for generating random numbers is the one used to generate standard uniform random numbers. Users are referred to standard sources for theoretical background. *LIMDEP*'s default random number generator is the L'Ecuyer's (1999) method. The specific generator used is his MRG32K3A multiple recursive generator. This generator has been shown to have excellent properties and has a period of about 2^{191} draws before recycling. The specific method used is as follows:

```
Define:       norm   = 2.328306549295728e-10,
              m1     = 4294967087.0,      m1     = 4294944443.0,
              a12    = 140358.0,          a13n   = 810728.0,
              a21    = 527612.0,          a23n   = 1370589.0,
Initialize    s10    = the seed,          s11    = 4231773.0,
              s12    = 1975.0,            s20    = 137228743.0,
              s21    = 98426597.0, s22    = 142859843.0.
```

Setting the seed for the generator is done by initializing $s10$ at the desired value and the remaining five values at the values shown. The six values constitute the seed for the generator, but to simplify the process, we chose the five values above, according to L'Ecuyer's recommendations, and the user or the program needs only to set $s10$. Now, the generator which produces u = one draw from U(0,1) is:

```
p1 = a12*s11 - a13n*s10, k = int(p1/m1), p1 = p1 - k*m1
if p1 < 0, p1 = p1 + m1, s10 = s11, s11 = s12, s12 = p1;
p2 = a21*s22 - a23n*s20, k = int(p2/m2), p2 = p2 - k*m2
if p2 < 0, p2 = p2 + m2, s20 = s21, s21 = s22, s22 = p2;
u = norm*(p1 - p2) if p1 > p2,
  = norm*(p1 - p2 + m1) otherwise.
```

The alternative RNG provided in *LIMDEP* is the Mersenne Twister. This generator was developed in 1997 by Makoto Matsumoto and Takuji Nishimura. It has been employed recently in numerous packages. The documentation is much too opaque to be laid out here. Details can be found in the authors' original article (Matsumoto and Nishimura (1998)). Random numbers drawn from other nonuniform populations are produced by transformations of the U(0,1) values, as discussed below.

R4A.4 Standard Normal Random Number Generation

Standard normal values are obtained using a method by Marsaglia. Let $u1$ and $u2$ be two standard uniform draws. The L'Ecuyer method noted above is used to obtain $u1$ and $u2$. Then, z = one draw from N(0,1) is obtained as follows:

```
x1 = u1+u1-1, x2 = u2+u2-1, s = x1*x1+x2*x2;
if s ≥ 1, get two new draws and start over;
v = log(s), v = sqr(-(v+v)/s);
z = x1*v
```

R4A.5 Random Number Generation from Other Distributions

Let z denote a draw from the standard normal distribution and u denote a draw from the standard uniform distribution. Draws from the other distribution are created as follows:

Rnn(m,s)	$= m + s \times z$
Rng(m,s)	$= \text{Exp}(m + s \times z)$
Rnt(n)	$= z \,/\, \sqrt{(1/d)\sum_{i=1}^{d} z_i^2}$
Rnx(d)	$= (1/d)\sum_{i=1}^{d} z_i^2$
Rnf(n,d)	$= (1/n)\sum_{i=1}^{n} z_i^2 \;\; / \;\; (1/d)\sum_{i=1}^{d} z_i^2$
Rnc(0)	= $z1/z2$
Rna(a,b)	= uses an intrinsic IMSL subroutine
Rnf(n,d)	= [Rnx(n)/n] / [Rnx(d)/d],
Rni(a,c)	= a times a draw computed with an IMSL intrinsic subroutine
Rne(q)	$= -q \times \log u$
Rnl(0)	$= \log [u/(1 - u)]$
Rnu(a,b)	$= a + u \times (b - a)$
Rnw(a,c)	$= (1/a)[-\log(u)]^{1/c}$
Rnh(a,b)	$= a - b\log(-\log(u))$
Rnp(m)	$= i$ such that $\sum_{j=1}^{i} \Pr(j) \le u$ and $\sum_{j=1}^{i+1} \Pr(j) > u$
Rnd(d)	$= \text{Int}(\, d \times u + 1)$
Rnm(p)	= Int[log(u)/log(1-p) - 1]
Rns(0)	$= 1/z$
Rnb(n,p)	$= i$ such that $\sum_{j=1}^{i} \Pr(j) \le u$ and $\sum_{j=1}^{i+1} \Pr(j) > u$
Rns(0)	= *1/normal*

R4A.6 Sampling from the Truncated Normal Distribution

Let u denote a draw from the standard uniform distribution, and let L and U denote the lower and upper limits of truncation respectively. Then, the single draw on u is transformed by

$$z^* = \Phi^{-1}\{\Phi(L) + u \times [\Phi(U) - \Phi(L)]\}$$

so that z^* is a draw from the standard normal distribution truncated between L and U. For truncation only in the lower tail, $\Phi(U) = 1$.

R4A.7 Random Sampling from the Multivariate Normal Distribution

A random draw, $\mathbf{v}$, from the K-variate normal population with mean vector $\boldsymbol{\mu}$ and covariance matrix, $\boldsymbol{\Sigma}$, is obtained by using the L'Ecuyer or Mersenne Twister method detailed above to obtain a K variate normal draw, $\mathbf{u}$. Then, $\mathbf{v} = \boldsymbol{\mu} + \mathbf{A}\mathbf{v}$, where $\mathbf{A}$ is the Cholesky square root of $\boldsymbol{\Sigma}$; $\boldsymbol{\Sigma} = \mathbf{A}\mathbf{A}'$.

R4A.8 Sample Variances

Throughout *LIMDEP*, sample variances are always computed in two passes using the sum of squared deviations, not the mean square minus the square of the mean. Thus,

$$V(x) = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2$$

is always computed by computing the mean first, then going back and computing the sum of squared deviations.

R5: Panel Data and Data for Discrete Choice Models

R5.1 Estimation Using Panel Data

There are many routines and estimators in *LIMDEP* that operate on panel data sets, i.e., those consisting of multiple rows of data per observation. This chapter describes the calculations and instructions needed to inform the program of the configuration of the data set. Descriptions of the specific model commands will extend these general parameters where needed for the particular application.

R5.2 Programs that Use Panel Data

Nearly every estimation program supported by *LIMDEP* supports a form of the model for panel data. Some of these are:

- regression – fixed and random effects and random coefficient models,
- binary logit and probit – fixed and random effects models,
- ordered probit and logit models with fixed and random effects and coefficients,
- tobit – fixed and random effects models,
- Poisson and negative binomial regressions – fixed and random effects models,
- stochastic frontier – fixed and random effects models,
- survival models – parametric models with time varying covariates,
- multinomial, multiperiod, random effects probit model,
- repeated observations, multiperiod random parameters logit model with random effects,

and the large number of programs that fit fixed effects, random parameters, and latent class models. These include the ones listed above as well as numerous others. The full list is roughly 50 different applications.

Panel data sets may be balanced or unbalanced. A *balanced panel* is one in which the group size, T_i, is the same for all i. An unbalanced panel has a varying group size. You can conveniently use either, but the second requires a bit of manipulation to be able to define the nature of the panel for estimation purposes.

No estimator in *LIMDEP* requires panels to be 'balanced' – the balanced panel is the special case. The program assumes that all panels are unbalanced. But, the set of observations must be 'contiguous.' That is, for all panel data models, the set of observations for a particular individual (group) must be a consecutive set of observations in the data set.

NOTE: Much of the econometrics literature on panel data models focuses on the balanced panel case and treats the unbalanced panel as in inconvenient extension. This is what is necessary to keep the mathematics manageable. (See, e.g., Baltagi (2005).) However, this a point at which theory and practice diverge. In *LIMDEP* all panels are treated as unbalanced. The balanced panel is the special case, though only in a trivial way that will be invisible to you.

R5.3 Panel Data Arrangement

Your estimation command for a panel data model must provide some means of determining the nature of the panel and how many observations are in a group. There is some variation across estimators that is discussed in detail with the specific descriptions of the models in the chapters to follow. But, most of these will use one of the conventions described here, or ones similar to them.

When the number of observations is fixed for each observation, as in **TSCS**, the command will generally include the specification **; Pds = T** as in

```
PROBIT      ; Lhs = y ; Rhs = x ; Pds = 5 $.
```

NOTE: Many of these models (e.g., probit, logit, tobit) will usually be estimated with data sets with one observation per individual. In that case, you must *omit* the **; Pds = 1** which would apply. The presence of **; Pds = anything** in the command usually does more than just provide a count; it invokes an altogether different estimation program. In *LIMDEP*, a cross section *is generally not* a panel with one observation per individual.

When the number of observations varies by individual *LIMDEP* requires you to provide a variable which gives the number of rows for that observation, in each row of the observation. For example, suppose your data consist of a panel of two individuals. The first has three observations (periods), the second has two. This data set has five rows, which could appear as

y	x	ni
4	2	3
5	0	3
2	5	3
7	1	2
3	9	2

The group size variable, *ni*, is then provided as the Pds identifier. The command would generally appear like the one for the frontier model below,

```
FRONTIER    ; Lhs = y ; Rhs = one,x ; Pds = ni $
```

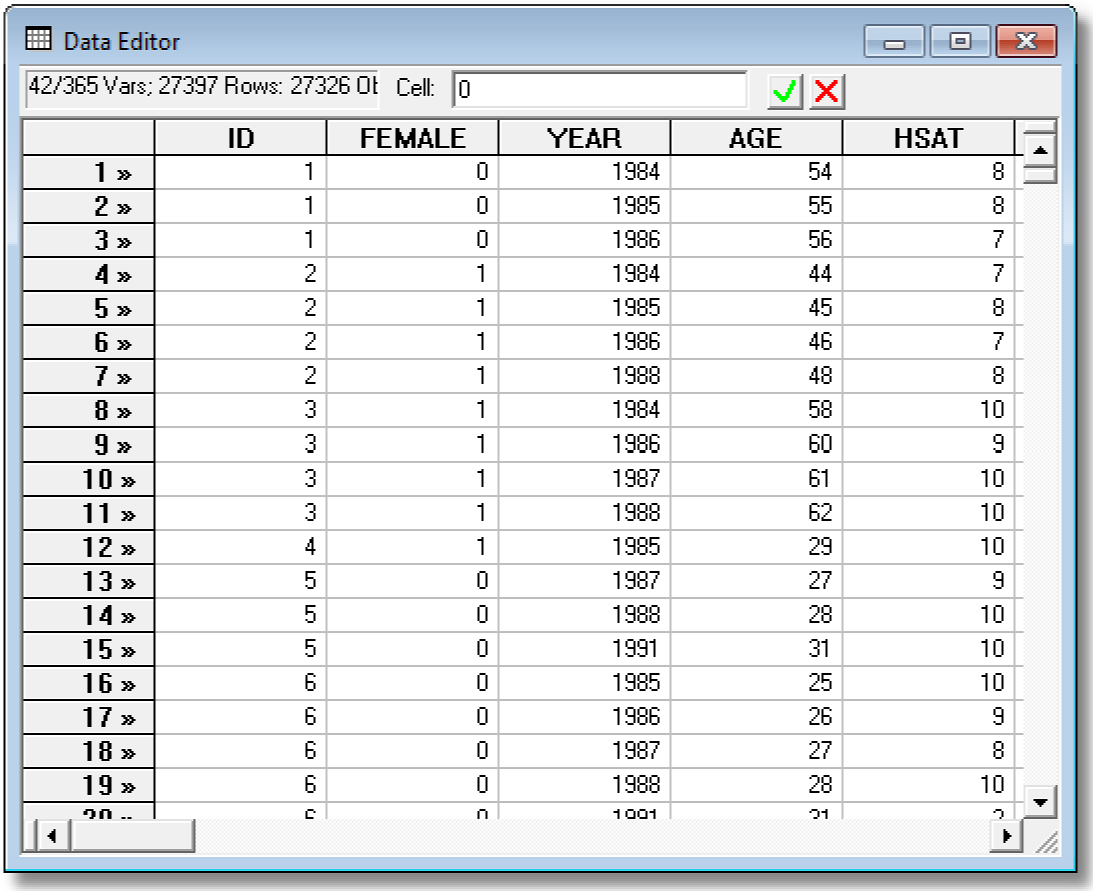
Estimators almost always allow either type of data set. Suppose, instead, that the first individual had two observations as well. The command might then be

```
FRONTIER    ; Lhs = y ; Rhs = one,x ; Pds = 2 $
```

The same arrangement is used in all of the other models except those fit by the **CLOGIT** command. The two elements of a panel data specification are this group count variable and a group indicator.

R5.3.1 Group Indicators and Within Group Observation Numbers

A stratification variable or group indicator is simply an indicator that shows *which* group an individual belongs to. Panel data sets generally contain this type of variable. Every unbalanced panel must have one or there would be no way to distinguish the groups. This is usually an ID variable of some sort, typically named *id* or *personid*. Figure R5.1 shows an example.



	ID	FEMALE	YEAR	AGE	HSAT
1 »	1	0	1984	54	8
2 »	1	0	1985	55	8
3 »	1	0	1986	56	7
4 »	2	1	1984	44	7
5 »	2	1	1985	45	8
6 »	2	1	1986	46	7
7 »	2	1	1988	48	8
8 »	3	1	1984	58	10
9 »	3	1	1986	60	9
10 »	3	1	1987	61	10
11 »	3	1	1988	62	10
12 »	4	1	1985	29	10
13 »	5	0	1987	27	9
14 »	5	0	1988	28	10
15 »	5	0	1991	31	10
16 »	6	0	1985	25	10
17 »	6	0	1986	26	9
18 »	6	0	1987	27	8
19 »	6	0	1988	28	10

Figure R5.1 Unbalanced Data Set with Household ID

The panel data estimators in the regression program use the stratification variable to construct group sizes and group means. The group sizes in the data set in Figure R5.1 are 3,4,4,1,3,5. The next section describes some commands for creating index, group size, and stratification variables. There are simple functions provided for creating group count variables from stratification indicators, and for creating a stratification variable when the group count is given, instead.

NOTE: In all cases where a stratification variable is used, *except the linear regression with fixed or random effects*, the stratification variable must take the values 1,2,...,N_g for some set of N_g groups. Commands for creating this variable are described below.

TIP: A frequently asked question concerns *LIMDEP*'s claim that a panel data set has an 'empty cell,' that is, a panel in which a group has no observations. A few estimators cannot proceed if this occurs. The problem is usually the values taken by the stratification indicator, and the most frequent cause is that the sample has been changed *after* the indicator was created. If you reduce your sample by rejecting observations or by skipping missing data, you may 'punch a hole' in your stratification indicator. Consider a sample consisting of strata [1,1,1,2,2,2,3,3,3,4,4,4]. If you set up your data, then give **SAMPLE ; 1-6, 10-12 $**, your remaining observations are [1,1,1,2,2,2,4,4,4], and the third cell is empty. The way to avoid this is to use the global setting described in Section R5.3.3 below. In general, you should

- Set the sample before creating your stratification indicator.
- Do not use **SKIP** with panels; use **REJECT** explicitly.

Some of the estimation programs described later, such as those in CLOGIT, have specific procedures for handling this situation. These are described in context. Moreover, most of the panel data model estimation programs handle missing data on their own, and you need not take any actions to deal with them. Again, this is discussed in context below.

In the example in Figure R5.1, the ID variable is the most convenient form of group indicator, consecutive integers. But, you might have some other form of identifier – for *LIMDEP*'s purposes, the indicator can be anything, so long as it is not the same for two consecutive groups. This could be something simple, such as a firm ID number, or something difficult such as a telephone number. All that is required is that the number be unique to the specific group and the same for all members of the group. Two functions are provided to create the type of indicator shown in the figure:

```
CREATE        ; id = Seq (identification variable) $
CREATE        ; id = Group Nmbr (period count variable) $
```

(Note that the second of these works with balanced panels as well. In each case, we will create

id = 1,1,1,…, 2,2,2,…, …, N,N,N…

For either case, as long as there is an ID variable, the command

```
CREATE        ;  id = Seq (identification variable) $
```

creates the unique, sequential group indicator. (Thus, for balanced panels which contain the ID variable, there are three functions that compute the group sequence ID.)

It is also useful to have an internal variable that indexes the observations in a group. This would be a variable that takes values 1,2,…,T_i within the group. These can be created using

Balanced panels:	**CREATE ; t = Trn (-T,0) $**
Unbalanced panels:	**CREATE ; t = Ndx (identification variable, 1) $**

TIP: The stratification variable used in these functions does not have to be sorted in the data. It is only an identifying code, and its actual numerical value and rank are not used. If you reset the sample after using Ndx, you will need to recreate the index variable.

The descriptive statistics program, regression model with fixed effects, the survival routines, the ordered probit model, and a few others use stratification variables directly. Where this information is needed, it is provided with the command specification

```
; Grp = name of the variable
```

NOTE: Previous versions of *LIMDEP* used **; Str = name of the variable** for this feature. You may still use that syntax.

R5.3.2 Group Size Variables for Panel Data

The group size variable or constant group size is used in all panel data estimators in *LIMDEP*. The general syntax is

```
Model       ; … ; Pds = group size variable $
```

There are different ways to create this variable for unbalanced panels. For balanced panels it is trivial:

```
CREATE      ; ni = 5 $
```

defines a panel data set with five observations for each individual. For unbalanced panels, you can use

```
CREATE      ; ni = Group Size (stratification variable) $
```

Group Size (id) works on any unique identifier within the panel, such as a person id, to create a variable that contains, within the group, the number of observations in the group. For example, suppose the panel contains two groups, one with three observations and one with two, and, initially, variables *personid*, *x*1 and *x*2.

personid	x1	x2	ti	time
1	3	13	3	1
1	9	22	3	2
1	8	14	3	3
2	4	9	2	1
2	0	11	2	2

The command

```
CREATE      ; ti = Group Size (personid) $
```

would create the variable *ti* shown above.

NOTE: This form of group count variable is used in all panel data estimators in *LIMDEP* to specify the panel. In earlier versions, an artificial regression command,

```
REGRESS      ; Lhs = one ; Rhs = one ; Str = personid ; Panel $
```

would be used to create the variable *_groupti* which would be identical to *ti* above. The regression form can still be used, but the preceding is likely to be simpler. In addition, in this version of *LIMDEP*, you can set the panel dimensions globally with a single '**SET**' command described in the next section, and the program will create the group count variables at the time they are needed.

The variable *time* shown above can be created internally with two functions

Group Nmbr (Pds variable)	= sequence number from 1 to T_i within a panel,
Seq (id variable)	= sequence number from 1 to T_i within a panel.

Seq and Group Nmbr produce the same result, but they are based on different input variables. Referring to the example above, *time* could be created using Group Nmbr (*ti*) or using Seq (*id*).

R5.3.3 Permanent Global Setting for Panel Data

Once you have the group identifier variable in place (or if it is part of the original data set), you can create a permanent setting for panel data that will free you from having to worry about the group count variable. Use

```
SETPANEL     ; Group = the identification variable
             ; Pds = name of a variable that the program will create $
```

After you set the panel in this fashion, you need only add **; Panel** to the commands that you use to fit panel data models. The very large advantage of this feature is that the group count variable is recreated at the time the model is fit. So, if you change the sample, it is not necessary to recompute the group count variable. The following example is based on the data set that appears in Figure R5.1. There are 7,293 observations in the full data set. The first model, a fixed effects probit model, uses the entire data set. Then, a second model is fit after removing from the sample all female headed households and all observations with *hsat* = 10. The first of these does not change the group count variable, since *female* is always the same through the panel. But, *hsat* varies over time, so by rejecting observations that have *hsat* = 10, we are reducing the sizes of some of the groups. The commands are

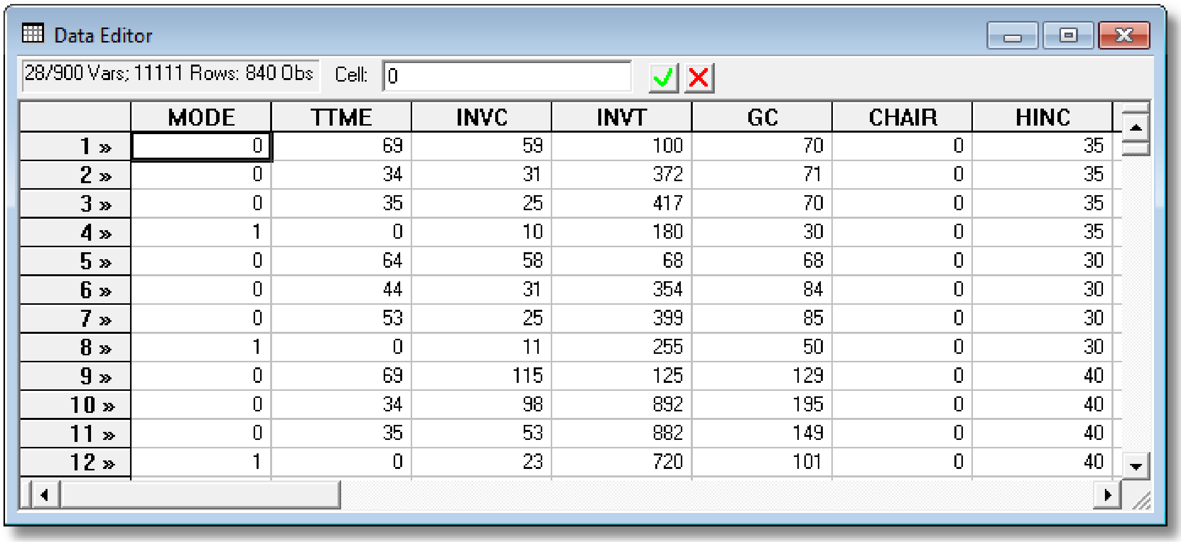
```
SETPANEL     ; Group = id ; Pds = grpti $
PROBIT       ; Lhs = public ; Rhs = one,age,educ ; Fem; Panel $
REJECT       ; female = 1 | newhsat = 10 $
PROBIT       ; Lhs = public ; Rhs = one,age,educ ; Fem; Panel $
```

The output is shown below. (Some of the results are not shown.)

```
-->PROBIT ; Lhs = public ; Rhs = one,age,educ ; Fem; Panel $
+------------------------------------------------------------------+
| Variable = ____________ Variable Groups    Max    Min   Average |
| GRPTI      Group sizes  ID         7293      7      1       3.7 |
+------------------------------------------------------------------+
-----------------------------------------------------------------------------
FIXED EFFECTS Probit Model
Dependent variable               PUBLIC
Log likelihood function     -1354.42890
Estimation based on N =  27326, K =1233
Inf.Cr.AIC  = 5174.858 AIC/N =     .189
Unbalanced panel has   7293 individuals
Skipped 6062 groups with inestimable ai
PROBIT (normal)  probability model
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  PUBLIC|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
     AGE|    -.06012***      .01027    -5.85  .0000     -.08025   -.03998
    EDUC|    -.30781***      .08242    -3.73  .0002     -.46934   -.14628
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
--> Reject ; female = 1 | newhsat=10$
--> probit ; Lhs = public ; Rhs = one,age,educ ; Fem; Panel $
+------------------------------------------------------------------+
| Variable = ____________ Variable Groups    Max    Min   Average |
| GRPTI      Group sizes  ID         3514      7      1       3.6 |
+------------------------------------------------------------------+
-----------------------------------------------------------------------------
FIXED EFFECTS Probit Model
Dependent variable               PUBLIC
Log likelihood function      -617.43315
Estimation based on N =  12504, K = 782
Inf.Cr.AIC  = 2798.866 AIC/N =     .224
Unbalanced panel has   3514 individuals
Skipped 2734 groups with inestimable ai
PROBIT (normal)  probability model
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  PUBLIC|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
     AGE|    -.11372***      .01607    -7.07  .0000     -.14522   -.08221
    EDUC|    -.24249**       .11312    -2.14  .0321     -.46420   -.02079
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

R5.4 Merging Invariant Variables into a Panel Data Set

Some panel data sets contain variables that do not vary across the observations in a group. A common example is the data for the CLOGIT (discrete choice) model used in numerous examples in this and the *NLOGIT* manual. The first 12 rows of this data set are shown in Figure R5.2. These data take the form of a panel (with four observations per person), in which the household income variable, *hinc*, is the same for each of the four rows. Some variables in the data set will be attributes of the choices, and, as such, will be different for each choice. Others may be characteristics of the individual, and will, therefore, be repeated on each record in the panel. *LIMDEP* allows you to keep separate data files for the variable and invariant data. This may result in a large amount of space saving. The data may be merged when they are read into *LIMDEP*, rather than in the original data set.



	MODE	TTME	INVC	INVT	GC	CHAIR	HINC
1 »	0	69	59	100	70	0	35
2 »	0	34	31	372	71	0	35
3 »	0	35	25	417	70	0	35
4 »	1	0	10	180	30	0	35
5 »	0	64	58	68	68	0	30
6 »	0	44	31	354	84	0	30
7 »	0	53	25	399	85	0	30
8 »	1	0	11	255	50	0	30
9 »	0	69	115	125	129	0	40
10 »	0	34	98	892	195	0	40
11 »	0	35	53	882	149	0	40
12 »	1	0	23	720	101	0	40

Figure R5.2 Panel Data with Invariant Variables

The command **MERGE**, which is similar to **READ** discussed in Chapter R3, will be used to combine two data sets. **MERGE** is used to interleave two files for a panel data set, in which one contains observations on variables that vary within a single 'group,' and a second contains variables that are only observed once for each individual. In a standard case, the larger file contains *T* observations for each of *N* individuals while the second contains one observation for each individual. In the merged data set, the values in the second data set are replicated as they are read.

There are two specifications that may be used to merge data. Both require a variable that is used to match the file to be expanded to the one that is already in memory.

R5.4.1 Using an ID Variable to Merge Data

Panel data sets typically have an ID number or some other identifying variable that is used to keep track of groups in the data set. For an example, the following description is provided for the data used in an application in Greene (2012)

```
Gary Koop and Justin L. Tobias, ‘Learning about Heterogeneity in Returns
to Schooling’, Journal of Applied Econometrics, Vol. 19, No. 7, 2004,
pp. 827-849.This panel data set consists of NT=17,919 observations from
N=2,178 individuals. The data are taken from the National Longitudinal
Survey of Youth. The data set is broken into two parts. The first part,
‘time_var.dat’, contains the time-varying characteristics together with the
individual-identification vector (denoted person_id). This file
contains17,919 observations on 5 variables. These variables are:
Column 1: Person_id (Ranging from 1-2,178).
Column 2: Education
Column 3: Log Hourly Wage
Column 4: Potential Experience
Column 5: Time Trend
The second part, ‘time_invar.dat,’ contains the time-invariant variables.
It contains 2,178 observations on 5 variables. These are:
Column 1: Ability
Column 2: Mother's Education
Column 3: Father's Education
Column 4: Dummy for Residence in Broken Home
Column 5: Number of Siblings
```

The syntax used to input such a data set is:

1. Read the original panel data set.

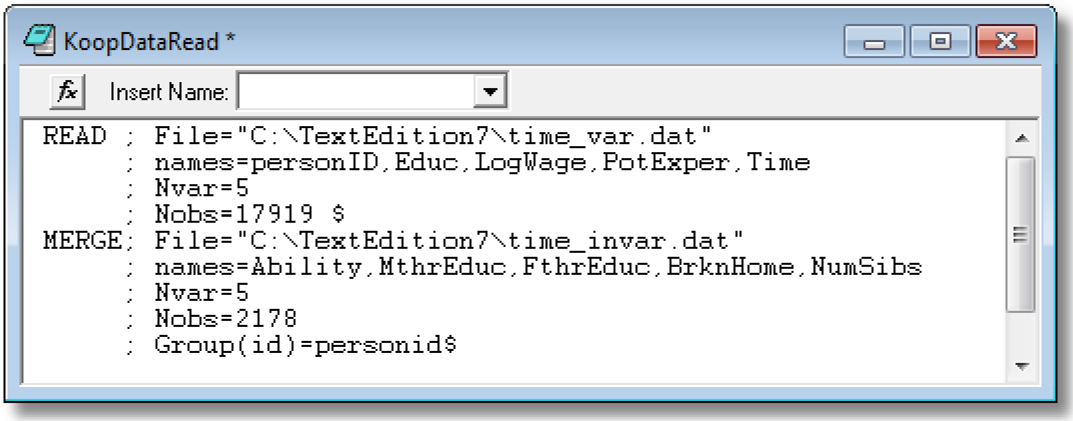
```
READ      ; File = var.dat
          ; Nobs = …  ; Nvar = …  ; Names = … $
```

2. Expand the invariant data.

```
MERGE     ; File = invar.dat
          ; Nobs = … ; Nvar = … ; Names = …
          ; Group(id) = ni $
```

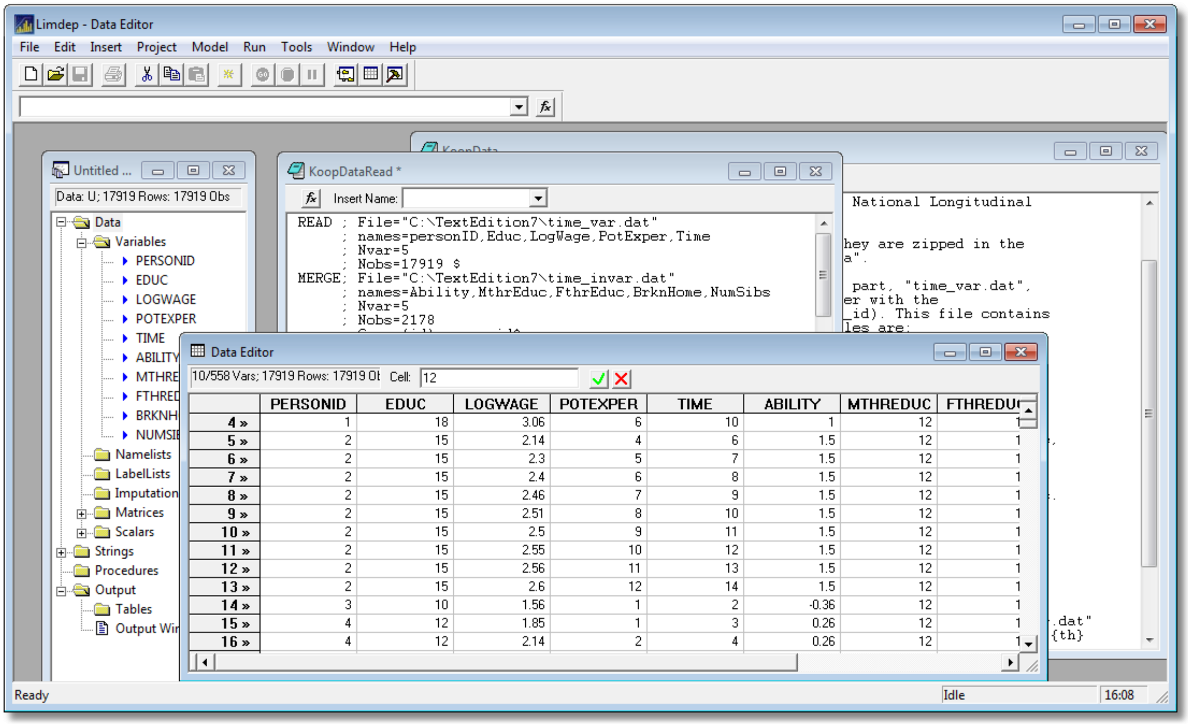
We downloaded the authors’ data from the *Journal of Applied Econometrics* website, extracted the two files and read them into *LIMDEP* as shown in Figure R5.3. The results are shown in Figure R5.4.

There are two restrictions on using **MERGE** to combine data sets. The procedure cannot be used with spreadsheet files (.xls) and it cannot be used with data sets arranged by variables (see Section R3.5.5). In addition, you cannot use **APPEND** to merge data sets. (See Section R3.10.)



```
READ  ; File="C:\TextEdition7\time_var.dat"
      ; names=personID,Educ,LogWage,PotExper,Time
      ; Nvar=5
      ; Nobs=17919 $
MERGE; File="C:\TextEdition7\time_invar.dat"
      ; names=Ability,MthrEduc,FthrEduc,BrknHome,NumSibs
      ; Nvar=5
      ; Nobs=2178
      ; Group(id)=personid$
```

Figure R5.3 Commands for Merging Time Varying and Time Invariant Data Sets



```
READ  ; File="C:\TextEdition7\time_var.dat"
      ; names=personID,Educ,LogWage,PotExper,Time
      ; Nvar=5
      ; Nobs=17919 $
MERGE; File="C:\TextEdition7\time_invar.dat"
      ; names=Ability,MthrEduc,FthrEduc,BrknHome,NumSibs
      ; Nvar=5
      ; Nobs=2178
```

Data: U; 17919 Rows: 17919 Obs

10/558 Vars; 17919 Rows: 17919 Ob Cell: 12

	PERSONID	EDUC	LOGWAGE	POTEXPER	TIME	ABILITY	MTHREDUC	FTHREDU
4 »	1	18	3.06	6	10	1	12	1
5 »	2	15	2.14	4	6	1.5	12	1
6 »	2	15	2.3	5	7	1.5	12	1
7 »	2	15	2.4	6	8	1.5	12	1
8 »	2	15	2.46	7	9	1.5	12	1
9 »	2	15	2.51	8	10	1.5	12	1
10 »	2	15	2.5	9	11	1.5	12	1
11 »	2	15	2.55	10	12	1.5	12	1
12 »	2	15	2.56	11	13	1.5	12	1
13 »	2	15	2.6	12	14	1.5	12	1
14 »	3	10	1.56	1	2	-0.36	12	1
15 »	4	12	1.85	1	3	0.26	12	1
16 »	4	12	2.14	2	4	0.26	12	1

Figure R5.4 Merged Data Sets

R5.4.2 Using a Group Count Variable to Merge Data

A second form of the key variable can be used if you have already created the type of group count variable used by *LIMDEP* in the panel data model estimation programs. The count variable gives group sizes for each observation in a group in a panel data set. For example, consider a panel with three individuals, and a variable number of observations per individual, two, then three, then two. The two data sets might look like

```
      File=var.dat
      Variable data
         x    y  ni
ind=1  1.1  4  2
       1.2  2  2
ind=2  3.7  8  3
       4.9  3  3
       5.0  1  3
ind=3  0.1  2  2
       1.2  5  2
```

```
File=invar.dat
Invariant data
          z
ind=1   100.7
ind=2    93.6
ind=3    88.2
```

Note the usual count variable, *ni*, for handling panels. To merge these files, use this setup

READ ; File = var.dat ; Nobs = 7 ; Nvar = 3 ; Names = x,y,ni $

This reads the original panel data set. Now, to expand the invariant data, the syntax is

MERGE ; File = invar.dat ; Nobs = 3 ; Nvar = 1 ; Names = z ; Group = ni $

The specification is the **; Group = ... specification**. The **; Group** specifies either a count variable, as above, or a fixed group size, as usual for *LIMDEP*'s handling of panel data sets. The resulting data will be

```
         x    y  ni    z
ind=1  1.1  4  2  100.7
       1.2  2  2  100.7
ind=2  3.7  8  3   93.6
       4.9  3  3   93.6
       5.0  1  3   93.6
ind=3  0.1  2  2   88.2
       1.2  5  2   88.2
```

Note the difference from the previous specification, where instead of **; Group = ni**, we used **; Group (id) = personid**. The '(id)' is the only difference between the two.

Checks and errors for this form of the command include:

- **; Nobs** must be given on the second **READ**.
- **; Nobs** must match exactly the number of groups in the existing data set.
- The existing panel must be properly blocked out by the *groups* variable or by a constant group size.

R6: Variable Lists and Labellists

R6.1 Namelists and Labellists

As part of estimation, it is necessary to define two sets of information, the variables to be used and the observations. *LIMDEP*'s data handling and estimation programs are written to handle large numbers of variables with simple, short commands. Two methods are provided to reduce the amount of typing involved in giving a list of names, the **NAMELIST** and a wildcard character. You can also define sets of text labels with the **CLIST** command. These are used to label displays of results that you compute with your own user written programs and to label the output of some descriptive routines such as histograms and cross tabulations.

R6.2 Lists of Variables in Model Commands

Lists of variables are used in every model estimation command and a large number of other commands, such as **WRITE**. Nearly all model commands are of the form

```
Model Command ; Lhs  = a variable
              ; Rhs  = a list of variables
              ; Rh2 = a list of variables $
```

Some model commands (such as **SURVIVAL**) have only the Lhs, others (such as **DSTAT** and **KERNEL**) only the Rhs, most have both Lhs and Rhs, **BIVARIATE PROBIT** has Lhs, Rh1 and Rh2, and **SURE** may have up to 50 lists for equations. Each of the lists may, in principle, have 150 or more names in it. As such, some shorthands will be essential.

R6.3 Wildcard Characters in Variable Lists

One simple shorthand for lists of variable names is the wildcard character, '*****.' You may use the '*****' character to stand for lists of variables in any variable list. There are three forms:

- * stands for all variables.

  LIST ; * $ requests a list of all existing variables.
  DELETE ; * $ is a global erasure of all data. (You should use **RESET**.)

- *aaaa** stands for all variables whose names begin with the indicated characters, any number from one to seven. For example: If you have variables *x*1, *x*2, *xa*,*xxx*,*xxy*,

  *x** = all five variables,
  *xx** = *xxx* and *xxy*,

  then the following command requests simultaneous scatter plots of all variables whose names begin with *x*,

  SPLOT ; Rhs = x* $

- *****_aaaa_ stands for variables whose names end with the indicated characters. For example, if you have *xa*, *ya*, and, *y*,

  REGRESS ; Lhs = y ; Rhs = one, *a $ regresses *y* on *one*, *xa*, and *ya*.

R6.4 Defining Namelists

The **NAMELIST** command defines a single name that is synonymous with a group of variables. It can be used in any model command and applies to the entire set of variables currently in the data array, regardless of how they got there. The command to define a namelist is

```
NAMELIST    ; name = list of variable names $
```

Several namelists may be defined with the same **NAMELIST** command by separating the definitions with semicolons, e.g.,

```
NAMELIST    ; w1 = x1,x2
            ; w2 = x3,x4,x5 $
```

For another example,

```
NAMELIST    ; job = butcher,baker,cndlmakr
            ; place = north,south,east,west
            ; person = job,place,income $
```

Note that in the example, the namelist *person* will contain eight variables, as the other two namelists are expanded and included with the eighth variable, *income*.

The lists of variables defined by separate namelists may have names in common. For example,

```
NAMELIST    ; w1 = x1,x2
            ; w2 = x2,x3 $
```

The restrictions on namelists are:

- The name must follow the usual rules for valid names.
- The list may not contain more than 150 names.
- The listed variables must already exist in the data set.
- A total of 25 namelists can be stored at any time.

R6.4.1 Combining and Shortening Namelists

Namelists may also contain the names of other namelists. This is a useful construction when you are building large models. For example:

```
NAMELIST    ; w1 = x1,x2
            ; w2 = x2,x3 $
NAMELIST    ; w12 = w1,x3
            ; ww = w1,w2 $
```

WARNING: Namelist definitions are 'static.' When a namelist definition contains another namelist, the full list of variables is expanded when the namelist is defined. This means that namelists are not updated when they are built up from other namelists, and these latter lists are changed. For example, if the preceding command were followed by **NAMELIST ; w3 = z,w2 $**, then the list in *w*3 would be *z*,*x*2,*x*3. But, if later, *w*2 were redefined to contain *c*1,*c*2,*q*, then *w*3 would still contain the variables *z*,*x*2,*x*3. It would not be updated to be consistent with the new definition of *w*2.

Note that *ww* contains the name of *x*2 twice. If you were to use this namelist in a model, you would find a problem of multicollinearity, as the same variable would appear twice. Some specific functions are provided to help you avoid this problem:

```
NAMELIST    ; name = OR (namelist1, namelist2, …) $
```

produces the *union* of the set of namelists given in parentheses;

```
NAMELIST    ; name = AND (namelist1, namelist2, … ) $
```

produces the *intersection* of the set of namelists given in parentheses;

```
NAMELIST    ; name = XOR (namelist1, namelist2, … ) $
```

produces the *exclusive union* of the namelists, that is, those variables that appear in the union, but not in the intersection. The exclusive union of the sets of variables is all those variables that appear in exactly one of the namelists.

Consider constructing a simultaneous equations model. Each equation contains some endogenous variables and some exogenous variables. For computing the two stage least squares estimator, you require the union of the sets of exogenous variables. Thus,

```
NAMELIST    ; x1 = one,x11,x12,x13,x14,z1,z2
            ; x2 = one,x11,     x13,    ,z1,    ,z3
            ; x3 = one,     x12,     x14,z1,       ,q $
NAMELIST    ; x = OR (x1,  x2,  x3) $
```

The last of these is the same as

```
NAMELIST    ; x = one,x11,x12,x13,x14,z1,z2,z3,q $
```

For convenience, you can delete a variable from a namelist with the syntax

```
NAMELIST    ; newlist = oldlist ~ variable $
```

The new list can be the same name as the old one, or a new one. For example, the variable *q* could be removed from the list *x* above with

```
NAMELIST    ; x = x ~ q $
```

R6.4.2 Deleting Namelists

If you run out of room for namelists, you can delete them with

```
NAMELIST    ; Delete name, name, ... $
```

Note that there is no semicolon between **Delete** and the names of the namelists being deleted. Also, you may delete more than one namelist in the command and you may delete and define namelists with a single command. For example,

```
NAMELIST    ; Delete states ; industry = agr,mfg $
```

You may also delete a namelist by selecting its name in the project window and pressing the Del key. In all these cases, you are only deleting the namelist definition, not the variables that are in the namelist. However, if you delete a variable that is contained in a namelist, then you are disabling every namelist that contains that variable. As a consequence, this does automatically delete the namelist. Consider the example in Figure R6.1. If we follow this definition by selecting the variable *ttme* in the project window and pressing Del, the following output results in the output window:

NAMELIST ; x = one,gc,ttme,invc,invt $
DELETE ; ttme $

```
Namelist X        is no longer defined.
```

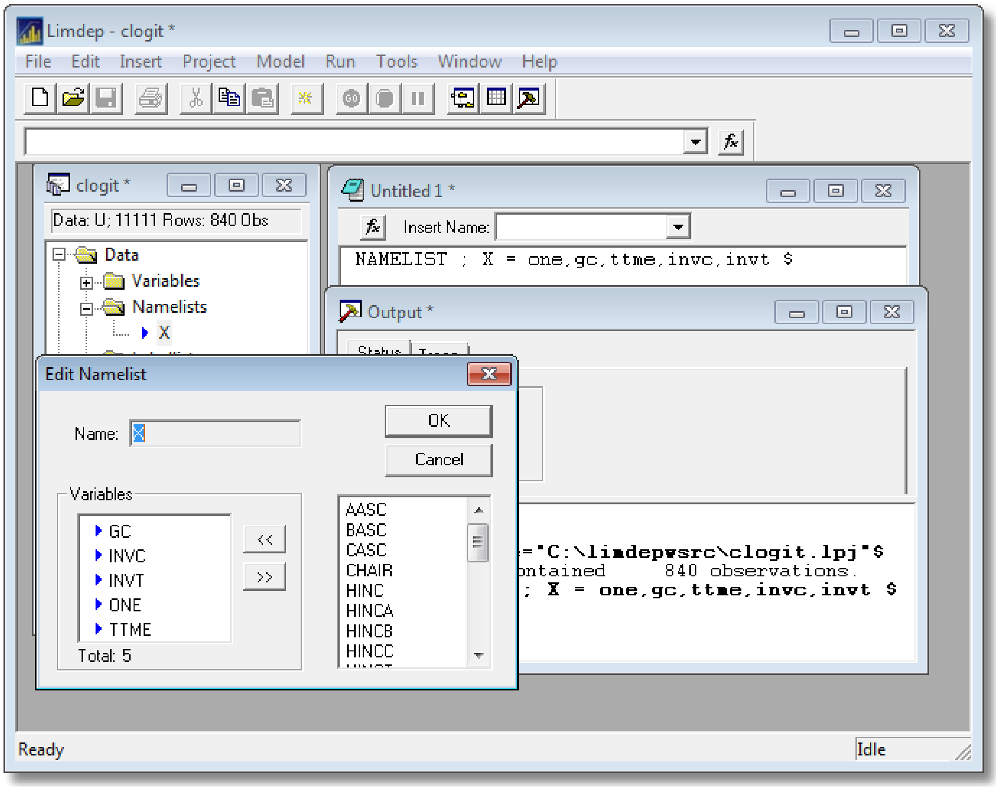
R6.4.3 Editing Namelists

By double clicking or right clicking the name of a namelist in the project window, you can enter an editor that allows easy modification of namelists. See Figure R6.1 for the setup.

You can also define new namelists with the New Namelist editor. There are several ways to reach this editor:

- Select New/Namelist from the Project menu,
- Select Item into Project/Namelist from the Insert menu,
- Right click the Namelists header in the project window, and select New Namelist.

All of these will invoke the dialog box shown in Figure R6.2.



Data: U; 11111 Rows: 840 Obs

```
NAMELIST ; X = one,gc,ttme,invc,invt $
```

Name: X

GC
INVC
INVT
ONE
TTME
Total: 5

AASC
BASC
CASC
CHAIR
HINC
HINCA
HINCB
HINCC

```
="C:\limdepwsrc\clogit.lpj"$
ontained      840 observations.
; X = one,gc,ttme,invc,invt $
```

Figure R6.1 Editing a Namelist

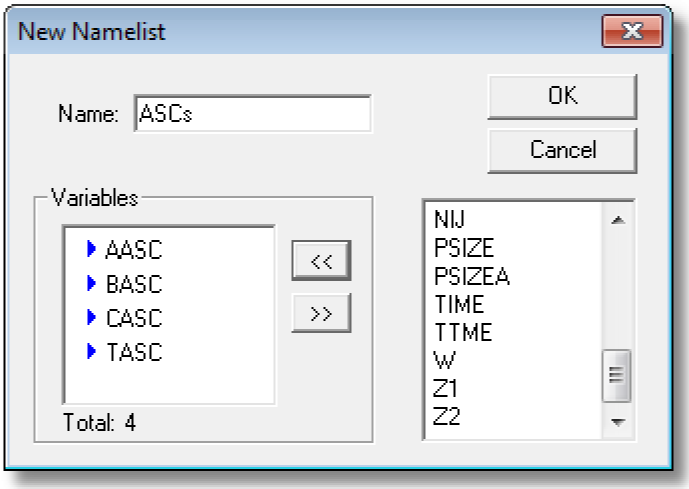


Figure R6.2 New Namelist Dialog Box

R6.5 Using Namelists

Namelists are used for several purposes. The primary uses are for defining variables in model instructions and in defining data matrices for MATRIX. They are also used for labeling statistical results from your estimation programs and for creating looping procedures that iterate over sets of variables.

R6.5.1 Using Namelists in Commands

The namelist is used in place of a list of names in a model command. For example, the following uses two namelists to set up an ordered probit model.

```
NAMELIST    ; demogrfc = age,sex,educ $
NAMELIST    ; family = haskids,married
OPROBIT     ; Lhs = hsat ; Rhs = one,demogrfc,family,income $
```

The namelist can be used in any setting that calls for a list of variables. For example, after the preceding,

```
LIST        ; family $
```

will produce a listing in the output window of the variables in family. The observations listed will be what is defined by the current sample. (See Chapter R7 for discussion of the current sample.) You can also use namelists with **WRITE**, as in

```
WRITE       ; family,demogrfc ; File = … <filename> … ; Format = CSV $
```

R6.5.2 Interaction Terms in Models and Partials

LIMDEP's new **SIMULATE** and **PARTIALS** programs will solve problems such as the following: Suppose the regression (or other type of) model is based on a function such as

$$y = \beta_1 + \beta_2 x + \beta_3 x^2 + \varepsilon.$$

Then, the partial effect of x on $E[y|x]$ is $\partial E[y|x]/\partial x = \beta_1 + 2\beta_2 x$. A natural way to estimate the model would be to use

```
CREATE      ; xsq = x*x $
REGRESS     ; Lhs = y ; Rhs = one,x,xsq $
```

However, if you request a set of partial effects for this model as stated, you will get one each for '*x*' and '*xsq*.' Neither is correct – the appropriate effect is a combination of the two. The program must somehow be informed that *xsq* is x*x. *LIMDEP* provides a way to do this by building the square (or interactions) visibly into the command. Thus, the better way to fit the model would be

```
REGRESS     ; Lhs = y ; Rhs = one,x,x*x $
```

Then

```
PARTIALS    ; Effects: x $
```

will provide the appropriate result. Building nonlinearities into model commands is discussed in Section R8.3, Chapter R11 and at several other points where model construction is developed. At this point, we note you can build those nonlinearities into the namelists that you use for model building. For the regression above, for example, the following would be useable:

```
NAMELIST    ; quadratc = one, x, x*x $
REGRESS     ; Lhs = y ; Rhs = quadratc $
```

This could then be followed by

```
PARTIALS    ; Effects: x $
```

which would produce the correct partial effect. (Note that **; Effects: quadratc** would be problematic.)

R6.5.3 Using Namelists in Matrix Algebra

The second major use of namelists is to define data matrices for matrix computations. This feature is shown in detail in Chapter R16, so we note it only briefly here. A namelist defines the columns of a data matrix. The current sample defines the rows. Thus, the following commands,

```
READ ; …       ; the clogit data set with 840 rows $
NAMELIST       ; x = one,gc,ttme,invc,invt $
MATRIX         ; xx = x'x ; invxx = <xx> ; b_ols = <x'x>*x'mode $
```

Compute an X’X, its inverse, and a least squares coefficient vector. The sample used is the 840 observations in the data set. If we now issue the command

```
SAMPLE         ; 1-200 $
```

Then the same **MATRIX** command will compute the three matrices using only the first 200 observations.

R6.5.4 Using Namelists to Display Model Results

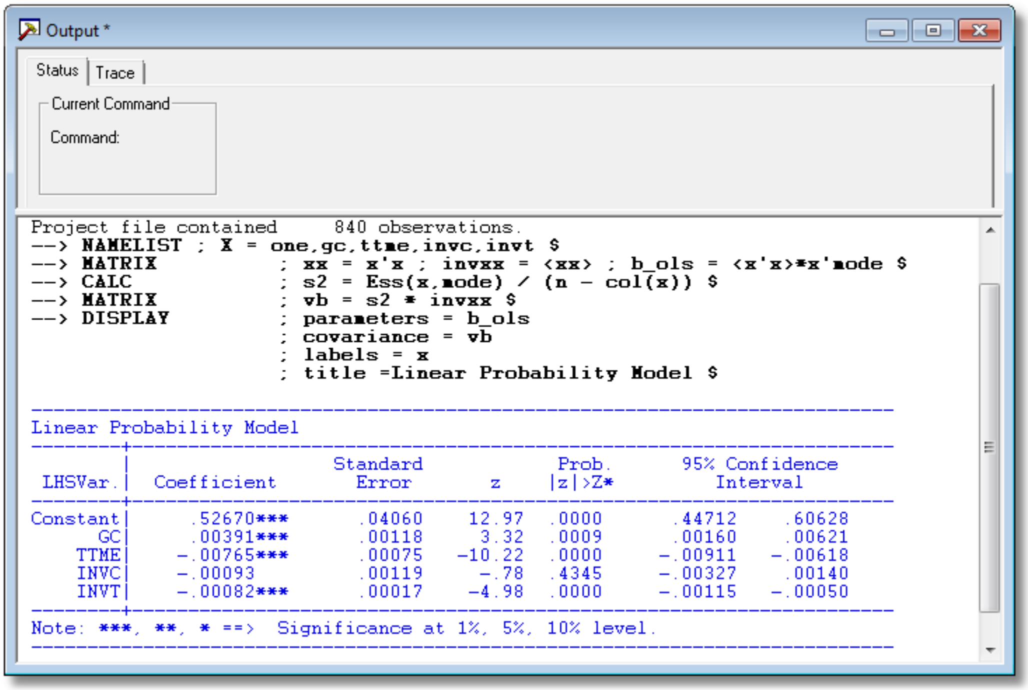
Namelists are used with the **DISPLAY** command to provide labels for statistical results. The following illustrates by continuing the earlier example:

```
NAMELIST       ; x = one,gc,ttme,invc,invt $
MATRIX         ; xx = x'x ; invxx = <xx> ; b_ols = <x'x>*x'mode $
CALC           ; s2 = Ess(x,mode) / (n – Col(x)) $
MATRIX         ; vb = s2 * invxx $
DISPLAY        ; Parameters = b_ols
               ; Covariance = vb
               ; Labels = x
               ; Title = Linear Probability Model $
```

In the **DISPLAY** command, the namelist is used in the **; Labels = x** to provide a set of names for the parameters that are to be shown. The results are shown in Figure R6.3. Note that the same numerical results would be produced by the following:

```
NAMELIST       ; x = one,gc,ttme,invc,invt $
REGRESS        ; Lhs = mode ; Rhs = x $
DISPLAY        ; Parameters = b
               ; Covariance = varb
               ; Labels = x
               ; Title = Linear Probability Model $
```

The matrices *b* and *varb* are automatically computed by the **REGRESS** command. The **DISPLAY** command will simply replicate the results produced (and shown) by the **REGRESS** command.



```
Project file contained     840 observations.
--> NAMELIST ; X = one,gc,ttme,invc,invt $
--> MATRIX            ; xx = x'x ; invxx = <xx> ; b_ols = <x'x>*x'mode $
--> CALC              ; s2 = Ess(x,mode) / (n - col(x)) $
--> MATRIX            ; vb = s2 * invxx $
--> DISPLAY           ; parameters = b_ols
                      ; covariance = vb
                      ; labels = x
                      ; title =Linear Probability Model $

-----------------------------------------------------------------------
Linear Probability Model
--------+--------------------------------------------------------------
        |               Standard            Prob.       95% Confidence
 LHSVar.|  Coefficient    Error       z    |z|>Z*          Interval
--------+--------------------------------------------------------------
Constant|      .52670***   .04060    12.97  .0000      .44712    .60628
      GC|      .00391***   .00118     3.32  .0009      .00160    .00621
    TTME|    -.00765***    .00075   -10.22  .0000    -.00911   -.00618
    INVC|    -.00093       .00119     -.78  .4345    -.00327    .00140
    INVT|    -.00082***    .00017    -4.98  .0000    -.00115   -.00050
--------+--------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------
```

Figure R6.3 Output Display Using Namelist for Variable Labels

R6.5.5 Using Namelists in CREATE

There are numerous computations done in **CALC** and **CREATE** that are based either on matrix algebra results or on treating an observation on a set of variables as a row vector. The **NAMELIST** definitions are essential for these computations. For an example, the following instructions create a variable *hii* that is computed as

$$hii_i = 1 - \mathbf{x}_i(\mathbf{X}'\mathbf{X})^{-1}\mathbf{x}_i'$$

where **X** is an $n\times K$ data matrix and $\mathbf{x}_i$ is the *i*th row of **X**. That is, $\mathbf{x}_i$ is the *i*th observation on the set of variables.

```
NAMELIST    ; x = the list of variables $
MATRIX      ; xxi = <x’x> $
CREATE      ; hii = 1 – Qfr(x,xxi) $
```

Namelists are also used to compute index functions. For example, the following commands compute the probabilities used to obtain the log likelihood function for a probit model

```
NAMELIST    ; x = one,gc,ttme,invc,invt $
PROBIT      ; Lhs = mode ; Rhs = x $
CREATE      ; probi = Phi((2*mode-1)*b’x) ; logp = Log(probi) $
CALC        ; List ; loglp = Sum(logp) $
```

The **CALC** command computes the log likelihood function by adding the observations contained in the variable *logp* that is calculated by the **CREATE** command. (The value of *loglp* is identical to that reported by the **PROBIT** command.)

R6.5.6 Using NAMELIST to Create a Data Matrix

NAMELIST may be combined with **CREATE** to create a template for a data matrix. The basic syntax is

```
NAMELIST    ; (new) ; listname = list of variables $
```

Both the *listname* and the variables must be new – they must not already exist. For example,

```
NAMELIST    ; (new) ; newz = znew1,znew2 $
```

The command shown does the following:

- Creates *znew*1 and *znew*2 as new variables
- Fills *znew*1 and *znew*2 with missing values (-999)
- Defines a new namelist, *newz*

This **NAMELIST** command is equivalent to two commands

```
CREATE      ; znew1,znew2 $
NAMELIST    ; znew = znew1,znew2 $
```

The **NAMELIST** command can be instructed to fill the new data matrix either with zeros instead of missing values, with

```
            ; (new = 0) …
```

or with random draws from the standard normal distribution with **; (new = N)** or with random draws from the standard uniform distribution with **; (new = U)**.

R6.5.7 Indexing Variables in Namelists

Variables in namelists may be *indexed* in any command in which they are used. The format is

listname: *index* to indicate the *i*th variable.

For example, in **NAMELIST ; x = yabc,ydef,y123 $**

x:1 is *yabc*,
x:2 is *ydef*, etc.

The index can be any numeric entity. For example,

```
CALC          ; i = 1 $
DSTAT         ; Rhs = x : i $
```

produces descriptive statistics for *yabc*. This construction can be extended to looping procedures. For a simple example,

```
NAMELIST      ; yvars = yabc,ydef,y123 $
PROC $
REGRESS       ; Lhs = yvars : i ; Rhs = one,x1,x2,x3 $
ENDPROC $
EXECUTE       ; i = 1,3 $
```

Variables, *yabc*, *ydef* and *y*123 are regressed in turn on *one*,*x*1,*x*2,*x*3.

R6.6 Labellists

A labellist is a list of text labels that you can use to label your results in several settings. The list is defined the same way a namelist is defined. The verb is **CLIST**. The command is

```
CLIST         ; labellistname = list of labels $
```

For example, to continue our example, we might define

```
CLIST         ; xvars = intrcept,gencost,termtime,invcost,invtime $
```

CLIST provides the two editing functions List and Delete.

```
CLIST         ; List clist name $
```

displays the list of labels. Note there is no semicolon after List. For example,

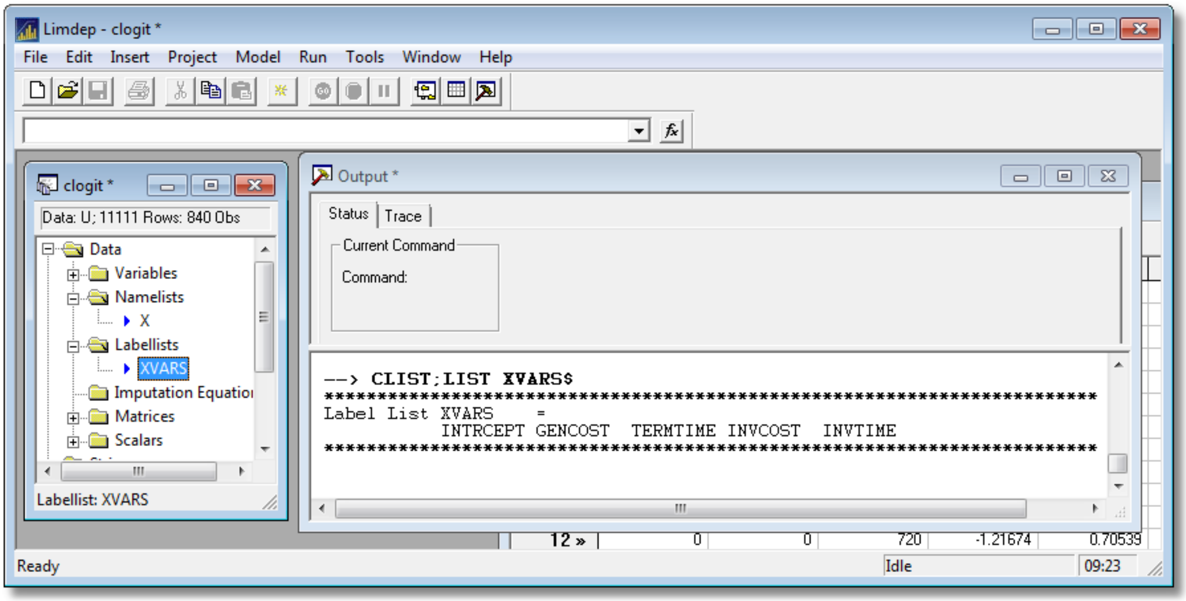
```
CLIST         ; List xvars $
```

will show the current contents of *xvars* in the output window. To delete a character list, use

```
CLIST         ; Delete clist name $
```

Once again, there is no semicolon before the name of the list to be deleted.

Labellists are part of the project and are displayed in the project window as shown in Figure R6.4. If you double click the name of a labellist in the project window, a listing of the contents of the list is shown in the output window.



```
--> CLIST;LIST XVARS$
****************************************************************************
Label List XVARS     =
           INTRCEPT GENCOST  TERMTIME INVCOST  INVTIME
****************************************************************************
```

Figure R6.4 Labellist in Project and Output Windows

The following shows the earlier example, using a character list rather than a namelist to provide the labels for the parameters. The display would be the same as shown in Figure R6.3 save for the parameter labels at the left of the output table.

```
DISPLAY     ; Parameters = b_ols
            ; Covariance = vb
            ; Labels = xvars
            ; Title = Linear Probability Model $
```

R7: The Current Sample and Missing Data

R7.1 The Current Sample

In most cases, you will read in a data set and use the full set of observations in your computations. But, it is quite common to partition the sample into subsamples and use its parts in estimation instead. You will also frequently want to partition the data set to define data matrices for use in the **MATRIX** commands.

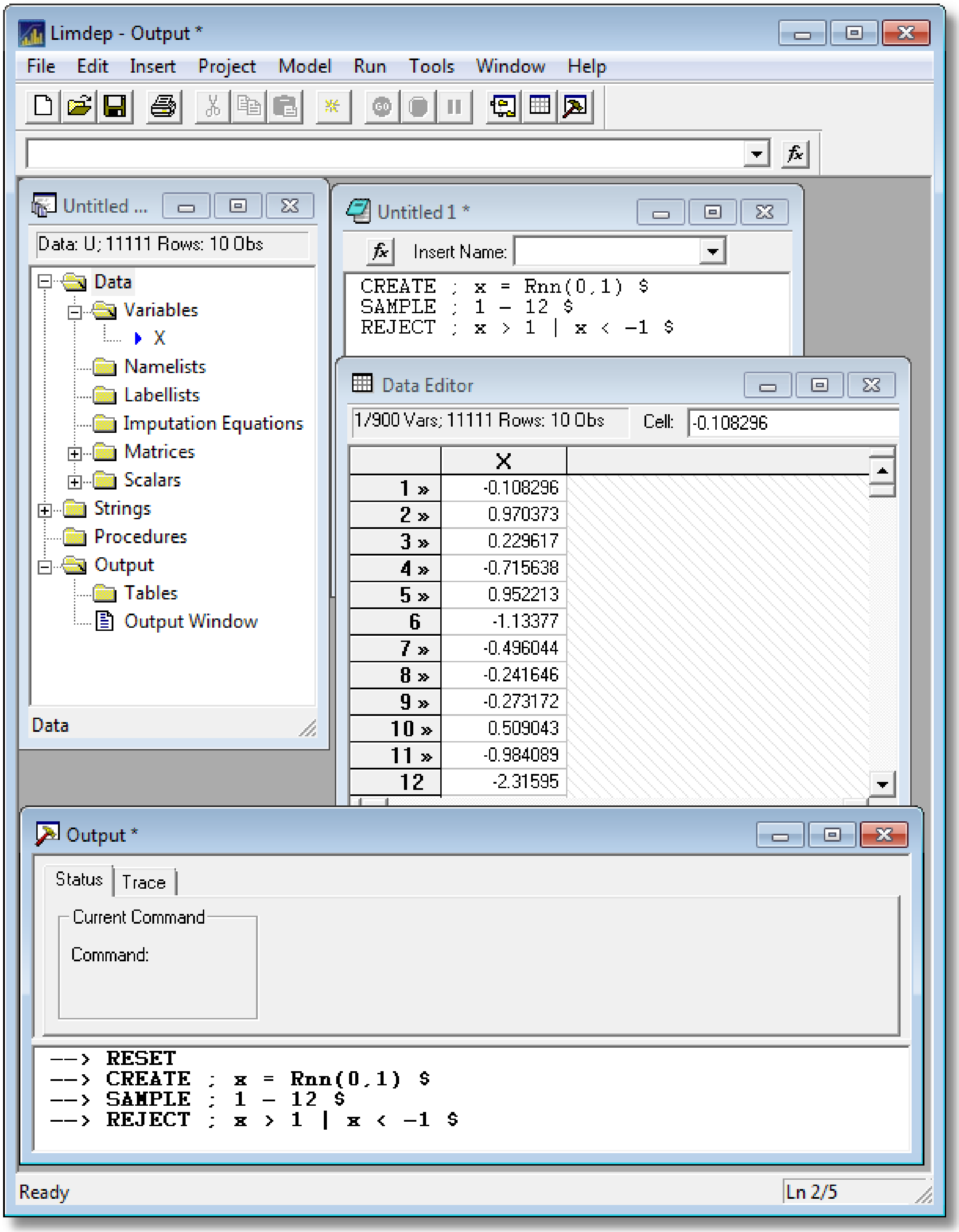
NOTE: The 'current sample' is the set of observations, either part or all of an active data set, which is designated to be used in estimation and in the data matrices for **MATRIX**, **CREATE**, etc.

The commands described in this chapter are used to designate certain observations either 'in' or 'out of' the current sample. With only a few exceptions, operations which use your data, such as model estimation and data transformation, operate only on the current sample. For example, if you have initially read in 10 observations on *x* and *y*, but then set the sample to include only observations 1-3, 6, and 8-10, nearly all commands will operate on or use only these seven observations. Thus, if you compute log(x), only seven observations will be transformed.

To define the current sample, *LIMDEP* uses a set of switches, one for each observation in the data set. Thus, when you define the sample, you are merely setting these switches. As such, the **REJECT** command does not actually remove any data from the data set, it merely turns off some of these switches. The data are not lost. The observations are reinstated with **SAMPLE ; All $**. Figure R7.1 shows the process. The sequence of instructions in the editing window creates a sample of draws from the standard normal distribution. The **SAMPLE** command chooses the first 12 of these observations, then the **REJECT** command removes from the sample observations that are greater than 1.0 or less than -1.0. This turns out to be observations 6 and 12, as can be seen in the data editor. The chevron to the right of the row number in the data editor is the switch discussed above.

There are two sets of commands for defining the current sample, one appropriate for cross section data and the other specifically for time series. Once the current sample is defined, you may further reduce it by random sampling observations from it. The **DRAW** command is used for this purpose. *LIMDEP* also provides methods of bootstrapping, which involve random sampling from the current sample *with replacement*.

NOTE: Section R7.5 discusses handling missing values in the data set. The missing values are taken to be part of the data set. Chapter R20 describes 'multiple imputation' procedures that are used to replace missing values with predictions from estimating equations that are built separately from a model that is being estimated. We will defer discussion of multiple imputation until after model setups are described in Chapters R8 and R9 and procedures are documented in Chapter R19. Multiple imputation methods rely on both of these.



```
CREATE ; x = Rnn(0,1) $
SAMPLE ; 1 - 12 $
REJECT ; x > 1 | x < -1 $
```

	X
1 »	-0.108296
2 »	0.970373
3 »	0.229617
4 »	-0.715638
5 »	0.952213
6	-1.13377
7 »	-0.496044
8 »	-0.241646
9 »	-0.273172
10 »	0.509043
11 »	-0.984089
12	-2.31595

```
--> RESET
--> CREATE ; x = Rnn(0,1) $
--> SAMPLE ; 1 - 12 $
--> REJECT ; x > 1 | x < -1 $
```

Figure R7.1 Current Sample and the REJECT Command

R7.2 Cross Section Data

Initially, observations are defined with respect to 'rows' of the data matrix, which are simply numbered 1 to whatever is the current setting of **ROWS**. (See Section R3.4 for the definition of **ROWS**.)

R7.2.1 Defining the Current Sample with the SAMPLE Command

Designate particular observations to be included in the current sample with the command

SAMPLE ; range, range, range, ..., range $

A 'range' is either a single observation number or a range of observations of the form lower-upper. For example,

SAMPLE ; 1, 12-35, 38, 44-301, 399 $

You can set the sample in this fashion, do the desired computations, then reset the sample to some other definition, at any time. To restore the sample to be the entire data set, use

SAMPLE ; All $

Because of the possibility of missing data being inadvertently added to your data set, *LIMDEP* handles this command as follows: 'All' observations are rows 1 to *N* where *N* is the last row in the data area which is not completely filled with missing data. In most cases, this will be the number of observations in the last data set you read. But, you can go beyond this last row by giving specific ranges on the command. For example, suppose you begin your session by reading a file of 100 observations. Thereafter, **SAMPLE ; All $** would be equivalent to **SAMPLE ; 1-100 $**. But, you could then do the following:

SAMPLE ; 1-250 $
CREATE ; x = Rnn(0,1) $ (random sample)

Now, since there are 250 rows containing at least some valid data, **SAMPLE ; All $** is equivalent to **SAMPLE ; 1-250 $**.

R7.2.2 Removing and Adding Observations with REJECT/INCLUDE

These commands are used to delete observations from or add observations to the currently defined sample. They have the form

VERB ; logical expression $

'**VERB'** is either **REJECT** or **INCLUDE**. 'Logical expression**'** is any desired expression that provides the condition for the observation to be rejected or included. It may include any number of levels of parentheses and may involve mathematical expressions of any complexity involving variables, named scalars, matrix or vector elements, and literal numbers. The operators are as follows:

Math and relational operators are +, -, *, /, ^, >, >=, <, <=, =, #.
Concatenation operators are & for 'and', | for 'or.'

A simple example appears in Figure R7.1. Another might be:

REJECT ; x > 0 $

For a more complex example, we compute an expression for observations which are not inside a ball of unit radius.

REJECT ; x^2 + y^2 + z^2 >= 1 $

For a third example with no obvious interpretation:

INCLUDE ; (r/s)*((c+7)*(x+2) * y^2 + z^3) > 1 | x +y < 0 $

The hierarchy of operations is ^, (*, /) (+,-), (>, >=, <, <=, =, #), &, |. Operators in parentheses have equal precedence and are evaluated from left to right. When in doubt, add parentheses. There is essentially no limit to the number of levels of parentheses. (They can be nested to about 20 levels.)

It is important to note that in evaluating expressions, you get a logical result, not a mathematical one. The result is either true or false. An expression which cannot be computed cannot be true, so it is false. Therefore, any subexpression which involves missing data or division by zero or a negative number to a noninteger power produces a result of false. But, that does not mean that the full expression is false. For example: $(x\ /\ 0) > 0 \mid x>y$ could be true. The first expression is false because of the zero divide, but the second might be true, and the 'or' in the middle returns 'true' if either expression is true. Also, we adopt the *C++*language convention for evaluation of the truth of a mathematical expression. A nonzero result is true, a zero result is false. Thus, your expression need not actually make logical comparisons. For example: Suppose *x* is a binary variable (zeros and ones). **REJECT ; x $** will reject observations for which *x* equals one, since the expression has a value of 'true' when *x* is not zero. Therefore, this is the same as **REJECT ; x # 0 $.**

REJECT deletes observations from the currently defined sample while **INCLUDE** adds observations to the current sample. You can use either of these to define the current sample by writing your command as

REJECT or INCLUDE ; New ; … expression … $

For a **REJECT** command, **; New** has the result of first setting the sample to all observations, then rejecting those observations which meet the condition specified in the expression. For an **INCLUDE** command, this has the effect of starting with no observations in the current sample and selecting for inclusion only those observations which meet the condition. In the latter case, this is equivalent to 'selecting cases,' as may be familiar to users of *SAS* or *SPSS*.

TIP: If your **REJECT** or **INCLUDE ; New** command has the effect of removing all observations from the current sample, *LIMDEP* takes this as an error, gives you a warning that this is what you have done, and ignores the command.

You may submit **REJECT** and **INCLUDE** commands from the dialog box shown in Figure R7.2. The dialog box is invoked by selecting Include or Reject in the Project:Set Sample menu or by right clicking in the data editor, clicking Set Sample, then selecting Reject or Include from the Set Sample menu. Note in the dialog box, the 'Reject observations from the current sample' option at the top is the **; New** specification in the command. Also, by clicking the query (?) button at the lower left, you can obtain information about these commands from the online Help file.

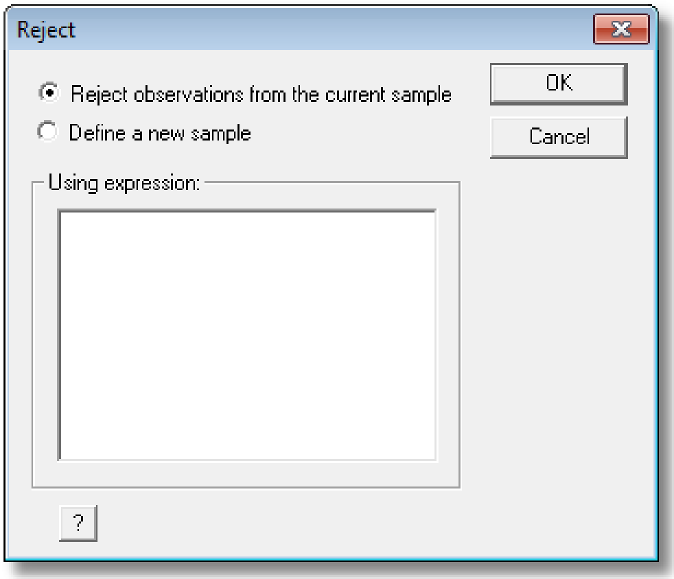
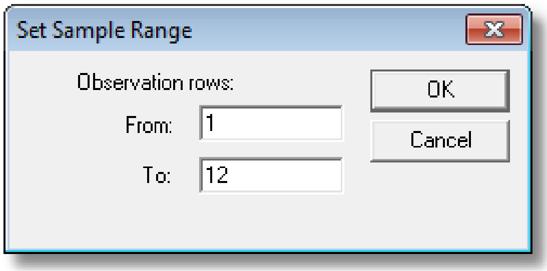


Figure R7.2 REJECT Dialog Box

The same Set Sample menu offers All, which just generates a **SAMPLE ; All $** command and Range which produces the dialog box shown in Figure R7.3.



Observation rows:
From: 1
To: 12

Figure R7.3 Set Sample Range Dialog Box

R7.2.3 Interaction of REJECT/INCLUDE and SAMPLE

REJECT and **INCLUDE** modifies the currently defined sample unless you include **; New**. But, **SAMPLE** always redefines the sample, in the process discarding all previous **REJECT**, **INCLUDE**, and **SAMPLE** commands. Thus,

```
      SAMPLE    ; 1-50,200-300 $
and   SAMPLE    ; 1-50 $
      SAMPLE    ; 200-300 $
```

are not the same.

Any of these three commands may appear at any point, together or separately. Before any appear, the default sample is **SAMPLE ; All $**.

TIP: If you are using lagged variables, you should reset the sample to discard observations with missing data *after you compute the lagged values*. This is generally not done automatically.

R7.2.4 Using Observation Tags to Set the Current Sample

The data set may include a set of alphabetic tags that are used to identify and group the observations. Observation tags are discussed in Section R3.2.2. For the example given there, the first 17 observations are yearly observations for Alabama, for which the observation tag is AL. The next 17 are for Arkansas, for which the tag is AR. Tags can be used to set the current sample as follows:

```
SAMPLE        ; [tagname] = string $
```

Thus

```
SAMPLE        ; [ST_ABB] = AL $
```

would be equivalent to

```
SAMPLE        ; 1 – 17 $
```

The **REJECT** and **INCLUDE** commands may be used the same way. For example,

```
REJECT        ; [ST_ABB] = AL $
```

would remove the 17 observations for Alabama from the sample if they were included at that point. Changing **REJECT** to **INCLUDE** in this command would add Alabama to the sample if it were not already included. In these three instructions, the “=” may also be “#” for “not equal,” thus reversing the direction of the change in the sample. For example,

```
SAMPLE        ; [ST_ABB] # AL $
```

would result in the sample being reset to observations 18 – 816.

R7.3 Time Series Data

When you are using time series data, it is more convenient to refer to rows of the data area and to observations by date, rather than by observation number. Two commands are provided for this purpose.

To give specific labels to the rows in the data area, use

```
DATES         ; initial date in sample $
```

The initial date may be one of:

Undated	same as before. (Use this to undo a previous **DATES** command.) **DATES ; Undated $**
YYYY	year for yearly data, e.g., **1951**. **DATES ; 1951 $**
YYYY.Q	year.quarter for quarterly data. Q must be 1, 2, 3, or 4. **DATES ; 1951.1 $**
YYYY.MM	year.month for monthly data. MM is 01 02 03 ... 12. **DATES ; 1951.04 $**

Note that .1 is a quarter, and .5 is invalid. The fifth month is .05, and the tenth month is .10, not .1. Once the row labels are set up, the counterpart to the **SAMPLE** command is

PERIOD ; first period - last period $

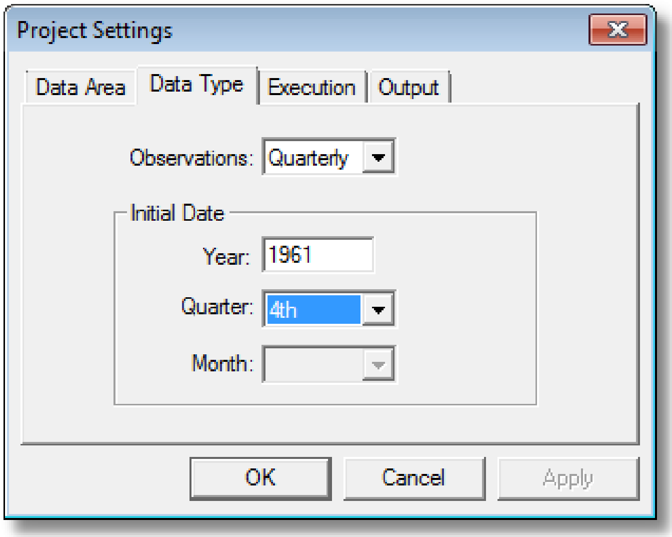
For example,

PERIOD ; 1964.1 - 1977.4 $

These two commands do not change the way that any computations are done with *LIMDEP*. They will change the way certain output is labeled. For example, when you use the data editor, the row markers at the left will now be the dates instead of the observation numbers.

NOTE: You may not enter a date using only two digits. Your dates must contain all four digits. No computation that *LIMDEP* does or command that you submit that involves a date of any sort, for any purpose, uses two digits. Therefore, there is no circumstance under which *LIMDEP* could mistake 20xx for 19xx. Any two digit date submitted for any purpose will generate an error, and will not be processed.

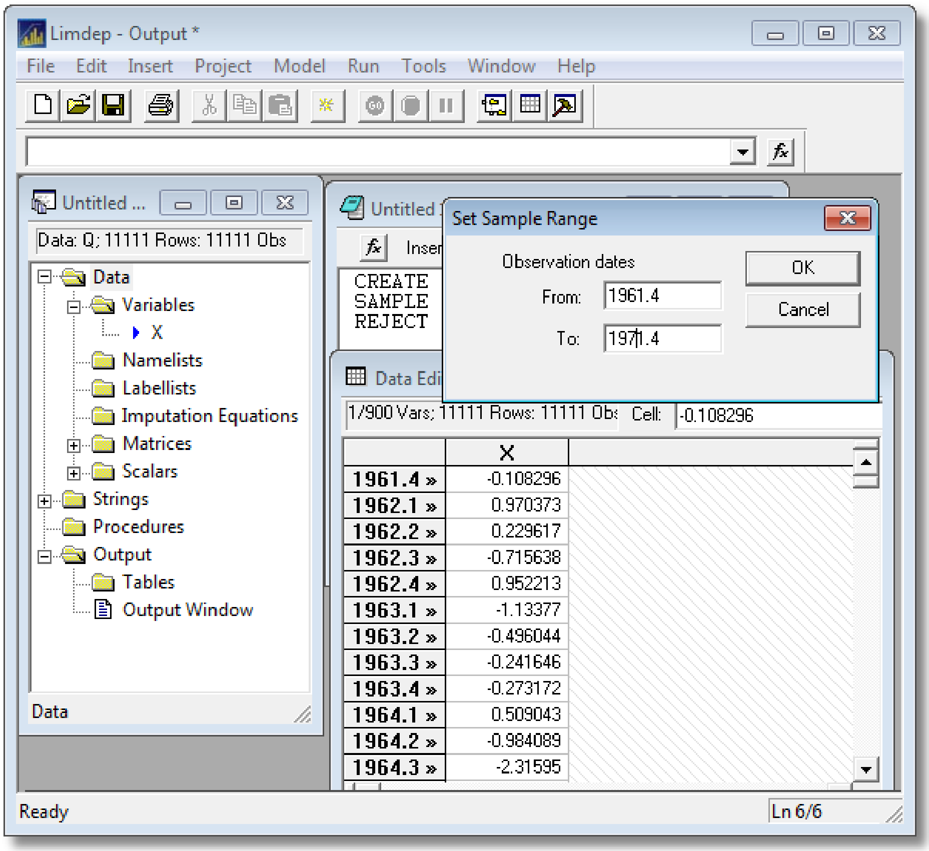
The **DATES** command may be given from the Project:Settings/Data Type dialog box, shown in Figure R7.4.



Observations: Quarterly
Year: 1961
Quarter: 4th
Month:

Figure R7.4 Dialog Box for the DATES Command

The **SAMPLE** and **PERIOD** commands may be given from the Project:Set Sample Range dialog box. See Figure R7.5.



Data: Q; 11111 Rows: 11111 Obs

```
CREATE
SAMPLE
REJECT
```

Set Sample Range

Observation dates

From: 1961.4

To: 1971.4

1/900 Vars; 11111 Rows: 11111 Obs Cell: -0.108296

	X
1961.4 »	-0.108296
1962.1 »	0.970373
1962.2 »	0.229617
1962.3 »	-0.715638
1962.4 »	0.952213
1963.1 »	-1.13377
1963.2 »	-0.496044
1963.3 »	-0.241646
1963.4 »	-0.273172
1964.1 »	0.509043
1964.2 »	-0.984089
1964.3 »	-2.31595

Figure R7.5 Dialog Box for the PERIOD Command

NOTE: The current data type, Data:U, Data:Y, Data:Q, or Data:M is displayed at the top of the project window. The data editor will also be changed to show the time series data. Figure R7.5 shows an example using quarterly data. The top of the project window displays the 'Q' which indicates quarterly data. The data editor has also automatically adjusted following the setting in Figure R7.4 for quarterly data beginning in 1961.4.

R7.4 Using the DRAW Command to Obtain Random Samples

You can draw a random sample from the current sample of observations with the **DRAW** command. This might be useful for bootstrap sampling, for example. (See Chapter R21 for applications and discussion.)

R7.4.1 Random Sampling from a Cross Section

The procedure is as follows: First, set the parent population to whatever is desired with **READ**, **SAMPLE**, **REJECT**, and **INCLUDE**. This results in *Nobs* observations. The command to draw a random sample is

DRAW ; N = number $

to sample '*number*' observations without replacement. *N* must be less than *Nobs*. Use

```
DRAW        ; N = number ; Rep $
```

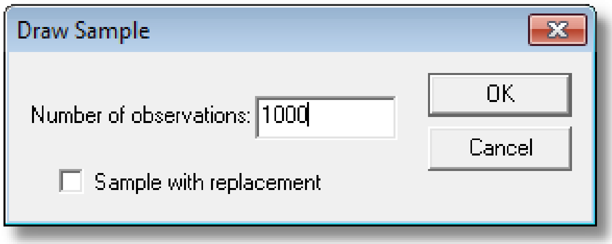
to sample with replacement. In this case, *Nobs* can be anything and *number* can be up to 100,000. For example:

```
SAMPLE      ; 1-100 $
CREATE      ; i = Trn(1,1) $  numbers from 1 to 100.
LIST        ; i $ will display numbers from 1 to 100 in order.
DRAW        ; N = 200 ; Rep $
LIST        ; i $ will display 200 random draws from i.
```

The original data are not changed, only the sample pointers are. Restore the original sample with

```
DRAW        ; N = 0 $
```

You can enter a **DRAW** random sample command dialog, as shown in Figure R7.6, by choosing Draw Sample from the Project Set Sample menu.



Number of observations: 1000

☐ Sample with replacement

Figure R7.6 Dialog Box for the DRAW Command

All commands which modify the sample turn off the random sample and restore the original data set. These are **REJECT**, **INCLUDE**, **SAMPLE**, **DATES**, **PERIOD**.

WARNING: Do not do any operation which modifies your existing data while this sampling procedure is in effect. The results will be unpredictable and can be severely problematic. This affects *all* operations that use the data.

WARNING: Do not use **SKIP** (see the next section) with bootstrapped samples or random samples. **SKIP** generates an internal **REJECT** command which will then automatically produce a **DRAW ; N = 0 $** command even if no observations get skipped.

R7.4.2 Random Sampling from a Panel Data Set

If you are using panel data and you want to sample randomly from the panel data set, the operation in the preceding section is probably not what you need. Assume for the moment that you have a balanced panel with, say, 1,000 individuals and five observations per individual for a total of 5,000 observations. If you use, say, **DRAW ; N = 1000 $**, you will draw a panel data set in which, now, some individuals will no longer have five observations. More likely, you would prefer to draw a sample of individuals from the original 1,000, so that the drawn sample is still a balanced panel of individuals randomly drawn so that each is still observed five times. In order to sample randomly from a panel in this fashion, use

```
DRAW        ; N = the number of individuals to draw
            ; Pds = either the fixed number of periods or the group size variable $
```

Note that the syntax allows you to draw a random sample of individuals from an unbalanced panel as well. Other parameters of this operation are:

- The full sample from which the sample is drawn may not be more than 500,000 observations.
- You may sample with replacement. The replacement is over individuals, not the individual observations.
- The bootstrapped sample may contain up to 20,000 groups in total.

R7.4.3 Simulating a Random Sample with Panel Data

You can simulate an unbalanced panel using the procedure described below. This uses some features that will be described in more detail in the chapters to follow, but this is a convenient place to introduce them. The procedure will create a sample for a balanced panel as well as an unbalanced one just by replacing the Rnd(*m*) in the first **CREATE** command with the fixed *m* that you want.

```
CALC        ; ni = ... the number of groups you want in your panel $
SAMPLE      ; 1 - ni $
CREATE      ; ti = Rnd(m) $  Set m to the largest group size you want.
MATRIX      ; mti = ti $
CALC        ; i1 = 1 ; i = 1 ; sumti = 0 $
PROCEDURE $
CALC        ; i2 = i1 + mti(i) – 1 $
SAMPLE      ; i1 - i2 $
CREATE      ; ... < the variables you want to simulate> ...
              ...   $
CREATE      ; groupti = mti(i) ; groupid = i $
CALC        ; sumti = sumti + mti(i) ; i1 = i1 + 1 ; i = i + 1 $
ENDPROCEDURE $
EXECUTE     ; N = ni $
SAMPLE      ; 1 -sumti $
```

You can now analyze these panel data. Use **; Pds = groupti** for group size counts. *Groupid* is a simple (1,2,...) group identifier.

R7.5 Missing Data

This section presents the information you will need to keep track of missing data when you operate *LIMDEP*. Section R7.5.5 describes the **SKIP** command, a particularly important device for handling missing data. Procedures for filling missing values with predictions from other models are described in Chapter R20.

R7.5.1 Reading Missing Data

When a data set contains missing values, you must indicate this in some way at the time the data are read. How you do this depends on the type of file you are reading:

Worksheet file from a spreadsheet program: Blank cells in a worksheet file are sufficient to indicate missing values. When *LIMDEP* writes a worksheet file for export, it, too will indicate missing data by a blank cell. It is not necessary to put any alphabetic indicator in the cell.

Formatted ASCII file: To indicate missing data in a formatted file, that is one that must be read with the **; Format = (...)** specification in the **READ** command, leave the fields blank. Then, add **; Blanks** to your **READ** command when you read the file.

Unformatted ASCII file: Any nonnumeric data in the field, such as the word 'missing' will suffice. Alternatively, a simple period surrounded by blanks will suffice. Note that in such a file, a blank will not be read as missing, since blanks just separate numbers in the data file.

CSV file: Missing values in a comma delimited CSV file are indicated by a single blank, which may appear as the first character in a line, between two commas in the middle of a line, or as a single blank after a comma at the end of a line. When you read such a file into *LIMDEP*, you must use **; Format = CSV** in your **READ** command.

DIF file: DIF files specifically contain alphanumeric data for missing values. Any nonnumeric value will suffice. **; Format = DIF** will pick these up appropriately.

Binary file: Missing data in a binary file must be indicated by the numeric value -999.

The internal code for a missing datum is -999. You may use this numeric value in any type of file to indicate a missing value. Upon reading the data, *LIMDEP* immediately converts any missing data encountered to the numeric value -999.

NOTE: The value -999 is *LIMDEP*'s default code for missing data. You can change this (very carefully) if necessary. Changing the missing value code is discussed in Section R3.2.8. In the text below, we will assume the code is -999. You should generally not need to change this.

R7.5.2 Missing Data in Transformations

Any transformation (see Chapter R4) that requires a value which turns out to be a cell containing missing data will return a missing value, not 0. Thus, if you compute *y* = Log(*x*), and some values of *x* are missing, the corresponding values of *y* will be also. Conditions are treated as follows: Suppose your transformation were **CREATE ; If (z = 5) y = Log(x) $**, and suppose for some observation, *z* is missing. If the variable named *y* already exists and this command is transforming *y*, then the condition would automatically be false, and *y* would not be set equal to Log(*x*), even if *x* were not missing. If this transformation is creating *y* for the first time, that is, if does not already exist, then the condition is, once again, automatically false, but now *y* is returned as the missing value, -999.

When computing a column of predictions, *LIMDEP* returns a missing value for any observations for which any of the variables needed to compute the prediction are missing, *even if the variable which will contain the predictions already exists at the time*. This results because when you request a model to produce a set of predictions, *LIMDEP* begins the process by ‘clearing’ the column in the data area where it will store the predictions. Data areas are cleared by filling them with the missing value code.

R7.5.3 Missing Data in Scalar and Matrix Algebra

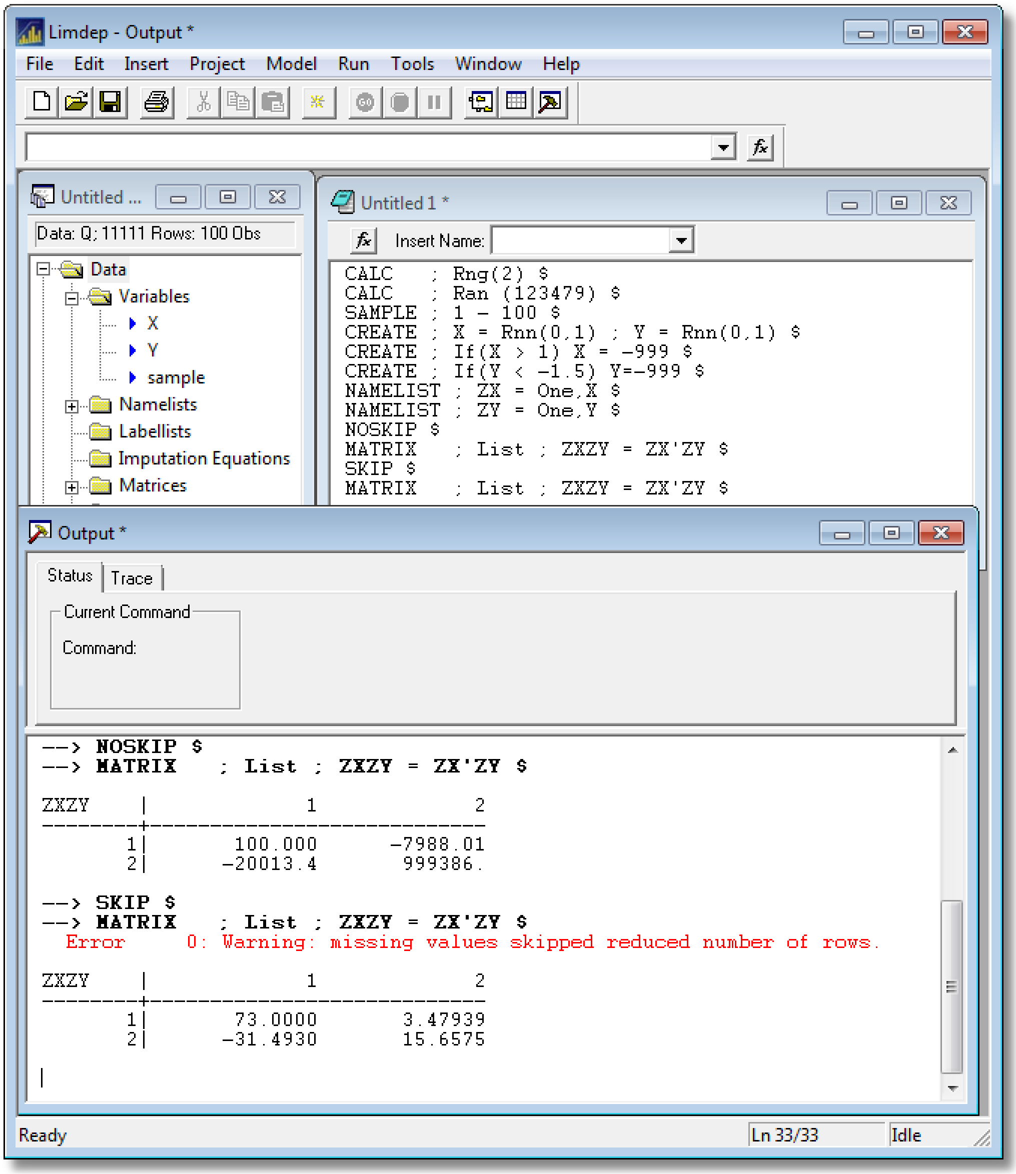
The treatment of missing values by **CALCULATE** is as follows:

- Dot products involving variables: The procedure is aborted, and -999 is returned.
- Max and Min functions: Missing data are skipped.
- Lik, Rsq, etc. (regression functions): Same as dot products.

The matrix algebra program that directly accesses the data in several commands, including **x’x**, for sums of squares and cross products, **<x’x>** for inverses of moment matrices, and many others will simply process them as if the -999s were legitimate values. Since it is not possible to deduce precisely the intention of the calculation, *LIMDEP* does not automatically skip these data or abort to warn you. It should be obvious from the results. You can specifically request this. If you do have the ‘**SKIP** switch’ set to ‘on’ (see Section R7.5.5) during matrix computations, *LIMDEP* will process **MATRIX** commands such as **x’x** and automatically skip over missing values. But, in such a case, the computation is usually erroneous, so your output will contain a warning that this has occurred, and you might want to examine closely the calculations being done to be sure it is really how you want to proceed.

Figure R7.7 illustrates the results discussed in the previous paragraph. The commands are shown in the editing window. (We have selected the Mersenne Twister RNG and set the seed explicitly so you can replicate the results.) Variables *x* and *y* are random samples of 100 observations from the standard normal distribution. The **CREATE** command changes a few observations in each column to missing values – the observations are not the same for *x* and *y*. The **NAMELIST** defines *zx* to be *x* and a column of ones – a two column matrix, and *zy* likewise. With **SKIP** turned on, the 2×2 matrix product **zx′zy** shows that there are 73 observations in the reduced sample (see the 73 that is **1′1** at the upper left corner) and a warning is issued. With **SKIP** turned off, in the first computation, the missing values are treated as -999s, and the resulting matrix has values that appear to be inappropriate.

There are also some **MATRIX** commands which return new variables, computed as linear combinations of existing variables. When missing data are encountered here, a missing value will be returned for the observation being computed, but no warning will be issued (as this might be deliberate).



```
CALC    ; Rng(2) $
CALC    ; Ran (123479) $
SAMPLE  ; 1 - 100 $
CREATE  ; X = Rnn(0,1) ; Y = Rnn(0,1) $
CREATE  ; If(X > 1) X = -999 $
CREATE  ; If(Y < -1.5) Y=-999 $
NAMELIST ; ZX = One,X $
NAMELIST ; ZY = One,Y $
NOSKIP $
MATRIX    ; List ; ZXZY = ZX'ZY $
SKIP $
MATRIX    ; List ; ZXZY = ZX'ZY $
```

```
--> NOSKIP $
--> MATRIX    ; List ; ZXZY = ZX'ZY $

ZXZY     |            1            2
---------+--------------------------
        1|      100.000     -7988.01
        2|     -20013.4      999386.

--> SKIP $
--> MATRIX    ; List ; ZXZY = ZX'ZY $
  Error     0: Warning: missing values skipped reduced number of rows.

ZXZY     |            1            2
---------+--------------------------
        1|      73.0000      3.47939
        2|     -31.4930      15.6575
```

Figure R7.7 Matrix Computations Involving Missing Data

R7.5.4 Missing Data in Estimation Routines

Unless you request it as described in the next section, *LIMDEP* will not account for the presence of missing data in estimation programs. That is, if the current sample contains rows of missing data, when you estimate a model or compute a moment matrix, the missing values will be included as if the value -999 were simply valid data. (See the first matrix in Figure R7.7.) This will, of course, seriously affect your results. Before using your data in estimation programs, you should use **REJECT** to delete observations which contain missing values.

TIP: If your estimator fails to converge, or the results look strange, or you get a diagnostic that the dependent variable is not coded correctly, you have probably failed to reject some observations which contain missing values. The descriptive statistics (means, standard deviations) will likely reveal some discrepancies.

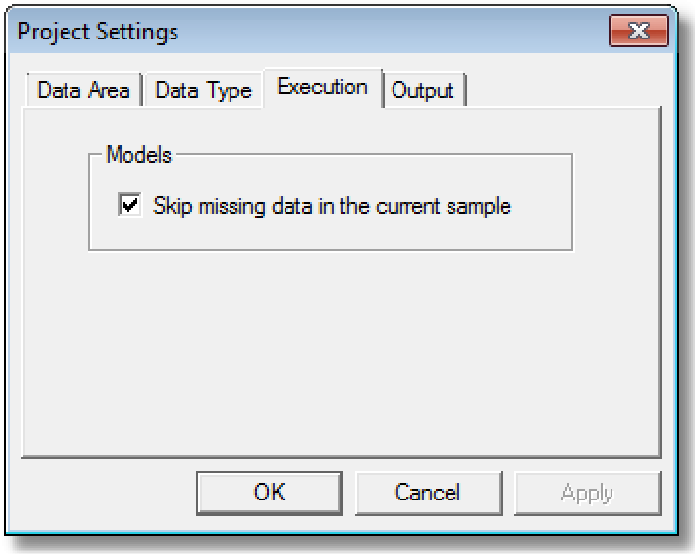
NOTE: **DSTAT**, the descriptive statistics command, automatically bypasses missing values. Descriptive statistics for each variable are computed separately, based only on the valid values for that variable. Covariances and correlations are based only on complete full rows of the data. The results show the resulting sample sizes. Most panel data estimators also bypass missing values, but most other estimation routines do not.

R7.5.5 Automatically Bypassing Missing Data – The SKIP Command

LIMDEP will skip missing data if you turn on the **SKIP** switch. This feature is controlled with the commands

SKIP (to turn it on) and **NOSKIP** (to turn it off).

SKIP can also be turned on and off with Project:Settings/Execution. See Figure R7.8.



☑ Skip missing data in the current sample

Figure R7.8 SKIP Switch from Project Settings/Execution

SKIP works as follows: At the time you give the command, the current sample is taken to be the 'master sample.' Note that this may or may not be the entire data set; the current sample may already be a subset of your data. With this setting 'on,' when you give a command to estimate a model, *LIMDEP* inspects *only the variables in the model command* and temporarily rejects observations for which any of these variables are missing. After the model is estimated, the sample is once again restored to the master sample.

For example: suppose the data consist of

Obs.	1	2	3	4	5	6	7	8	9	10	11	12
X	1	2	3	1	5	.	8	2	.	.	9	5
Y	.	8	2	.	1	3	4	5	6	7	6	1

The sequence of commands

```
REJECT      ; New ; x > 8 $
SKIP
REGRESS     ; Lhs = y ; Rhs = one,x $
```

First deletes observation 11. The master sample is then 1-10,12. The regression uses observations 2, 3, 5, 7, 8, and 12. After it is run, the current sample is restored to 1-10,12. Any subsequent **SAMPLE**, **REJECT**, **INCLUDE**, or **PERIOD** resets the master sample. Turn this feature off with the command **NOSKIP**.

NOTE: If you are fitting any of the following models with panel data (this is almost all of the models *LIMDEP* supports):

Probit with random or fixed effects
Logit with random or fixed effects
Ordered probability with random effects
Poisson with random or fixed effects
Negative binomial with random or fixed effects
Frontier with random effects
Tobit with random effects
Parametric survival models
CLOGIT estimator and all estimators in *NLOGIT* 6 (treat the NALT rows of data as a panel)
All fixed effects estimators specified with FEM
All random parameters estimators
All latent class estimators

LIMDEP will automatically bypass the full group of observations for any individual in a panel if any of the observations contain missing data on any of the variables that are being used to fit the model. You do not have to make any adjustments to enable this feature; it is handled internally. For these cases, you should *not* have the **SKIP** switch on. If you wish to restrict the sample before estimation in these settings, use **REJECT**.

R7.5.6 Nonlinear Optimization Programs and Using SKIP Generally

SKIP cannot operate on estimators that you define with **MAXIMIZE**, **MINIMIZE**, **GMME**, **NLSQ** or **NLSURE**. The reason is that when **SKIP** is turned on, *LIMDEP* inspects the data contained in lists of variables, **; Rhs**, **; Rh2**, **; Inst**, and so on. The function definitions in these commands do not contain lists; they contain variables intermingled with other entities such as parameters and scalars. You can request that **SKIP** be applied to a set of variables that you specify by adding

; Skip = any list of variables

to your command. You should include variables that appear in the function definitions but are not in explicit lists. (See Chapters E14, E25, E66 and E67 for discussion of nonlinear function optimizers.)

HINT: You may use this with any estimation command. It will usually be redundant in other models, but the feature is provided generally since we assume that we cannot anticipate every possible model specification or usage.

R7.6 The Sample Used for the Most Recent Model

When the sample contains missing values, and the gaps in the data set are for different observations for different variables, *listwise deletion* of observations can result in different samples being used for different models. After each model is estimated, a created variable named *_sample_* will contain a binary indicator of the observations used to fit the most recent model. One use for this variable might be for examination of the data used for that model. For example, suppose that *x*1,*x*2,*x*3 and *x*4 all have missing data for different observations.

```
SKIP
NAMELIST    ; x = x1,x2,x3,x4 $
REGRESS     ; Lhs = y ; Rhs = one,x $
DSTAT       ; If [ _sample_ = 1] ; Rhs = x $
```

will produce the statistics for the observations used in the regression. If the**; If […]** function were not included in the **DSTAT** instruction, all of the nonmissing observations for each variable would be used.

R8: Commands for Estimating Models

R8.1 Model Specifications of Variables and Weights

This chapter will describe the common form of all model estimation commands. Equation specifications are described in Sections R8.3-R8.7. Using weights in estimation is discussed in Section R8.8.

Chapters R9-R13 contain general discussions on the important statistical features of the model estimators, such as output of model results, interpreting results and obtaining partial effects. Chapters R11 and R12 also describe procedures that are generally used after a model is estimated, such as testing hypotheses, retrieving and manipulating results, and analyzing restrictions on model parameters. In terms of your use of *LIMDEP* for model estimation and analysis, Chapters R8-R12 are the most important general chapters in this part of the manual.

R8.2 Model Commands

All model commands are built from the basic form

```
Model Command ; Lhs = dependent variable
              ; Rhs = list of independent variables
              ; ... other parts specific to the model ; ... $
```

The 100 or so different models are specified by changing the model name or by adding or subtracting specifications from the template above. At different points, other specifications, such **; Rh2 = a second list**, are used to specify a list of variables. These will be described with the particular estimators in the *Econometric Modeling Guide*. Different models will usually require different numbers and types of variables to be specified in the lists above. You may always use namelists at any point where a list of variables is required. Also, a list of variables may be composed of a set of namelists.

NOTE ON CONSTANT TERMS IN MODELS: Of the over 100 different models that *LIMDEP* estimates, only one, the linear regression model estimated by stepwise regression, automatically supplies a constant term in the Rhs list. *If you want your model to contain a constant term, you must request it specifically by including the variable 'one' among your Rhs variables. You should notice this in all of our examples below*. The variable *one* is provided by the program; you do not have to create it. You can, however, use *one* at any point, in any model where you wish to have a constant term, and any **MATRIX** command based on a column of ones as a variable in the analysis of data.

ADVICE ON MODEL SPECIFICATION: It is fairly rare that a model would be explicitly specified without a constant. In almost all cases, you should include the constant term. Omitting the constant amounts to imposing a restriction that will often distort the results, sometimes severely. (We recall a startling exchange among some of our users in reaction to what appeared to be drastic differences in the estimates of a probit model produced by *LIMDEP* and *Stata*. The difference turned out to be due to the omitted constant term in the *LIMDEP* command.) In a few cases, though it is not mandatory by the program, you definitely should consider the constant term essential – these would include the stochastic frontier and the ordered probit models. However, in a few other cases, you should *not* include a constant term. These are certain fixed effects, panel data estimators, such as regression, logit, Poisson and so on. (In most cases, if you try to include an overall constant term in a fixed effects model, *LIMDEP* will automatically remove it from the list.)

In addition to the specifications of variables, there are over 200 different specifications of the form

```
; sss  [= additional information]
```

which are used to complete the model command. These specifications are specific to the model being estimated. In some cases, these are mandatory.

```
SETPANEL    ; Group = id ; Pds = ti $
REGRESS     ; Lhs = ... ; Rhs = ... ; Panel ; Random Effects $
```

This is the model command for a random effects linear regression model. The last specification is necessary in order to request the particular panel data model. Without the last two specifications, the command simply requests linear least squares. Another example

```
REGRESS     ; Lhs = … ; Rhs = … ; Heteroscedasticity $
```

Requests a robust, heteroscedasticity corrected covariance matrix. In other cases, specifications will be optional, as in

```
REGRESS     ; Lhs = ... ; Rhs = ... ; Keep = yf $
```

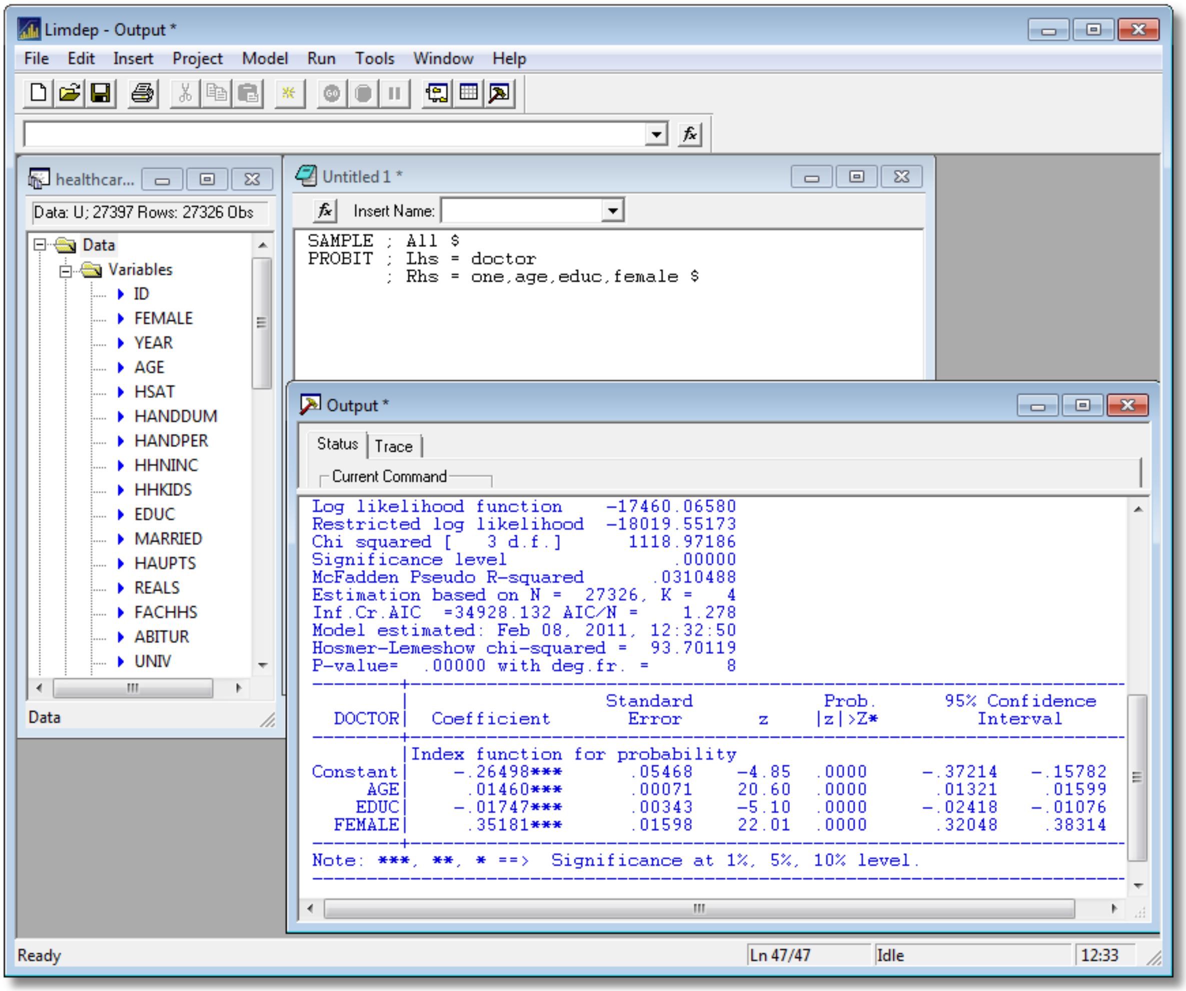
which requests *LIMDEP* to fit a model by linear least squares, then compute a set of predictions and keep them as a new variable named *yf*. The regression is computed regardless, **; Keep = name** is added to request the additional step of putting the predictions in the data area as a new variable named *yf*.

Some model specifications are general and are used by most, if not all, of the estimation commands. For example, the **; Keep = name** specification in the command above is used by all single equation models, linear or otherwise, to request *LIMDEP* to keep the predictions from the model just fit. In other cases, the specification may be very specific to one or only a few models. For example, the **; Cor** in

```
SWITCHING REGRESSION ; Lhs = ... ; Rh1 = ... ; Rh2 = ... ; Cor $
```

is a special command used to request a particular variant of the switching regression model, that with correlation across the disturbances in the two regimes. The default is to omit **; Cor**, which means no correlation.

Figure R8.1 shows a model command and the resulting output window using the basic form of model command.



```
SAMPLE ; All $
PROBIT ; Lhs = doctor
       ; Rhs = one,age,educ,female $
```

```
Log likelihood function     -17460.06580
Restricted log likelihood   -18019.55173
Chi squared [   3 d.f.]      1118.97186
Significance level              .00000
McFadden Pseudo R-squared      .0310488
Estimation based on N =  27326, K =   4
Inf.Cr.AIC  =34928.132 AIC/N =    1.278
Model estimated: Feb 08, 2011, 12:32:50
Hosmer-Lemeshow chi-squared =  93.70119
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                  Standard            Prob.       95% Confidence
  DOCTOR|  Coefficient       Error        z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|   -.26498***      .05468     -4.85  .0000      -.37214    -.15782
     AGE|    .01460***      .00071     20.60  .0000       .01321     .01599
    EDUC|   -.01747***      .00343     -5.10  .0000      -.02418    -.01076
  FEMALE|    .35181***      .01598     22.01  .0000       .32048     .38314
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

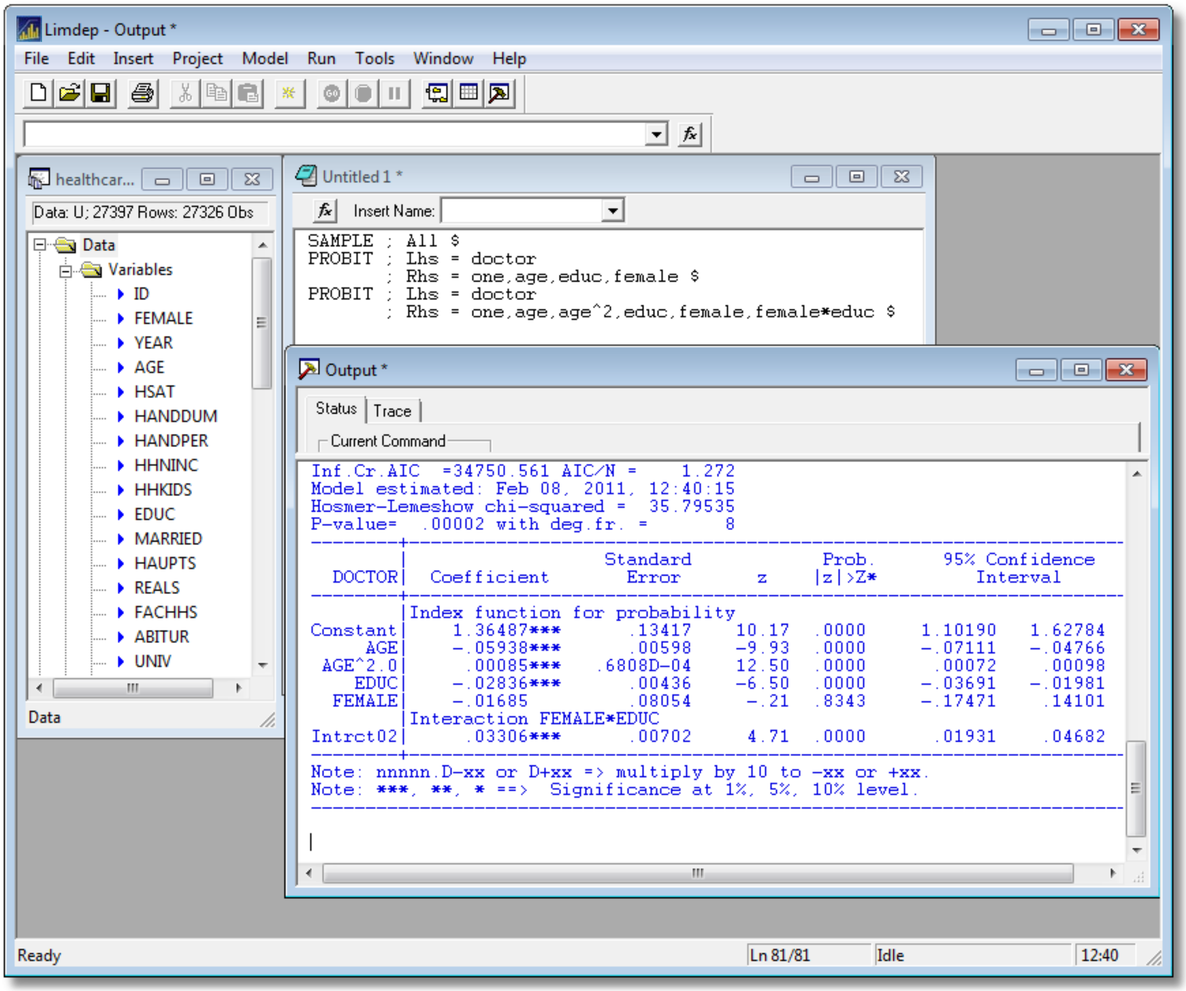
Figure R8.1 Basic Model Estimation Command and Results

R8.3 Interaction Terms and Nonlinear Functions of Variables

Models are usually specified with interaction terms, squares of variables, and so on. In general, it is necessary to create these variables separately in the data set and then include the transformed variables in the model. To continue the example above, a model that includes the square of age and different impacts of education by gender would be obtained by adding

```
CREATE    ; agesq = age * age  ;  fem_educ = female * educ $
PROBIT    ; Lhs = doctor ; Rhs = one,age,agesq,educ,female,fem_educ $
```

You can include these transformed variables directly in the model command, rather than first adding them to the data set. Figure R8.2 shows the command and results.



```
SAMPLE  ; All $
PROBIT  ; Lhs = doctor
        ; Rhs = one,age,educ,female $
PROBIT  ; Lhs = doctor
        ; Rhs = one,age,age^2,educ,female,female*educ $
```

```
Inf.Cr.AIC  =34750.561 AIC/N =    1.272
Model estimated: Feb 08, 2011, 12:40:15
Hosmer-Lemeshow chi-squared =  35.79535
P-value=  .00002 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient     Error        z    |z|>Z*        Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|   1.36487***     .13417    10.17  .0000     1.10190   1.62784
     AGE|   -.05938***     .00598    -9.93  .0000     -.07111   -.04766
  AGE^2.0|    .00085***   .6808D-04  12.50  .0000      .00072    .00098
    EDUC|   -.02836***     .00436    -6.50  .0000     -.03691   -.01981
  FEMALE|   -.01685        .08054     -.21  .8343     -.17471    .14101
        |Interaction FEMALE*EDUC
Intrct02|    .03306***     .00702     4.71  .0000      .01931    .04682
--------+--------------------------------------------------------------------
Note: nnnnn.D-xx or D+xx => multiply by 10 to -xx or +xx.
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Figure R8.2 Estimating a Model with Interaction Terms

R8.3.1 Interaction Terms and Logs of Variables in Commands

Any term in a model command may appear as follows:

variable1 * variable2	such as female * educ
variable1 * variable2 ^ power	such as educ * age^2
variable1 / variable2	such as gdp / pricelvl
variable1 / variable2 ^ power	such as income / price^2
variable1 ^ power	such as age^2
log(variable)	such as log(wage)

Note variable1 may be same as variable2, so *age*age* is the same as *age*^2. These variables may appear in any list in a model, that is, **; Rhs**, **; Lhs**, **; Rh1**, **; Rh2**, and so on. The only restrictions are that only the forms above are supported and the exponent in the power functions must be positive and of the form n, .n or n.n. That is, they must be explicit numbers. A square root, for example is obtained with power = .5. Note as well, the formulation also allows you to do division. There is a potential for complications here. You will be protected from dividing by zero. However, the result of your attempt to divide by zero will be to return a missing value. Since this calculation is being done 'on the fly' during manipulations of the data, it is not possible to stop execution and take some corrective action. A zero divide can turn an observation with no missing values into one in which a variable in the model does contain a missing value. You should be careful at the outset if you will use this feature, and ensure that you will not be dividing by zero at any time. There is also a second possible (less probable, however) complication in this procedure. In general, the procedure will not allow you to raise a negative number to a power. In principle, it is possible to raise a negative number to an integer power, but *LIMDEP* is not checking for this possibility. An attempt to raise a negative number to a nonzero power produces a missing value. Finally, the log function returns a missing value if the variable is not positive.

In addition to the obvious convenience of streamlining the model commands and making it unnecessary to compute the additional variables in your data set, this new feature provides a significant capability to work with the **PARTIAL EFFECTS** command described in Chapter R11. Consider a probit model with the square of age in it. The probability is

$$\text{Prob}(doctor = 1|\mathbf{x}) = \Phi(\beta_1 + \beta_2 educ + \beta_3 female + \beta_4 age + \beta_5 age^2).$$

The partial effect of *age* in this model is

$$\partial\Phi(.)/\partial age = \phi(\ \beta_1 + \beta_2 educ + \beta_3 female + \beta_4 age + \beta_5 age^2) \times (\beta_4 + 2\beta_5 age).$$

If we request partial effects in the usual way with a simple probit command

```
CREATE      ; agesq = age^2 $
PROBIT      ; Lhs = doctor ; Rhs = one, educ, female, age, agesq
            ; Partial Effects $
```

The response is

```
-----------------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with
respect to the vector of characteristics
Average partial effects for sample obs.
--------+--------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
  DOCTOR|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
    EDUC|    -.00571***     -.10285    -4.58  .0000     -.00816   -.00327
  FEMALE|     .13036***      .09924    22.38  .0000      .11894    .14177   #
     AGE|    -.02186***    -1.51297   -10.10  .0000     -.02610   -.01762
   AGESQ|     .00031***     1.00074    12.66  .0000      .00026    .00036
--------+--------------------------------------------------------------------
#  Partial effect for dummy variable is E[y|x,d=1] - E[y|x,d=0]
Elasticity for a binary variable is marginal effect/Mean.
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

This looks convincing, but it is not what we wanted. The program has computed a separate partial effect for *age* and *agesq*. The problem is that the program has no way of knowing that *agesq* is the square of *age*; it could be anything. It is just another variable in the equation. (This is the point made by the now famous paper of Ai and Norton (2003).)

The **PARTIAL EFFECTS** feature described in Chapter R11 makes use of the in line interaction and nonlinear functions and computes the partial effects correctly. For the example shown, if we change the commands to

PROBIT ; Lhs = doctor ; Rhs = one, educ, female, age, age^2 $
PARTIAL EFFECTS ; Effects: age $

the results are

```
--> PARTIALS ; Effects: age $
---------------------------------------------------------------------
Partial Effects  Analysis for Probit Probability Function
---------------------------------------------------------------------
Effects on function with respect to AGE
Results are computed by average over sample observations
Partial effects for continuous AGE      computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dAGE            Partial    Standard
(Delta method)     Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
Partial effect      .00467     .00024   19.30      .00420      .00514
```

The in line statement of the relationship between *age* and *age* squared enables the partial effects program to compute the appropriate derivatives.

NOTE: This new capability is built into nearly all the models that are fit by *LIMDEP*, including the linear regression model, probit, tobit, and dozens of other.

In any statistical command (**DSTAT, REGRESS, SURE**, and so on), logs of any variables may be specified directly in the command instead of being created beforehand. For example, the commands

CREATE ; ly = Log(y) ; lx = Log(x) $
DSTAT ; Rhs = ly, lx $
REGRESS ; Lhs = ly ; Rhs = one, lx $

could be replaced with the commands

DSTAT ; Rhs = Log(y), Log(x) $
REGRESS ; Lhs = Log(y) ; Rhs = one, Log(x) $

Remember, though, in the second case, the logs of the variables are not kept in your data area; they are retained only for the current regression or model.

R8.3.2 Interaction Terms and Nonlinear Terms in Namelists

Namelist definitions may contain any number of interactions, logs, and so on, as defined in the preceding section. For example,

```
NAMELIST    ; translog = one, lnx1, lnx2, lnx1^2, lnx2^2, lnx1*lnx2 $
```

might be used for convenience to specify a production function. Then,

```
REGRESS     ; Lhs = Log(y) ; Rhs = translog $
```

could be used to estimate the model. It is important to note a difference between placing interactions in a namelist and in a model command. When interactions are placed in your model command, they replace the internal table that defines the constructed variables. Each model command defines its own new set of variables. The list is replaced by the next model command. But, **NAMELIST** defines a permanent set of definitions, since the namelist, itself is permanent. To continue the example, the commands,

```
NAMELIST    ; cobbdgls = one, lnx1, lnx2 $
NAMELIST    ; translog = one, lnx1, lnx2, lnx1^2, lnx2^2, lnx1*lnx2 $
REGRESS     ; Lhs = Log(y) ; Rhs = translog $
CALC        ; lt = logl $
REGRESS     ; Lhs = Log(y) ; Rhs = cobbdgls $
CALC        ; lc = logl ; chisq = 2*lt-lc) $
```

will set up a likelihood ratio test of the null Cobb-Douglas model against the alternative translog model.

R8.3.3 Managing Constructed Variables in the Data Set

Namelists that contain interaction terms are accumulated in a table. This set of information is part of the project you are working on. When you save your project, the namelist definitions are saved in it, so the definitions will be intact when you reload your project.

The accumulated table accounts for cases when namelists use the same constructions. For example, the following creates three namelists.

```
SAMPLE      ; 1-1000 $
CREATE      ; x1 = Rnu(0,1) ; x2 = Rnu(0,1) ; x3 = Rnu(0,1) $
CREATE      ; lnx1 = Log(x1) $
CREATE      ; lnx2 = Log(x2) $
NAMELIST    ; cobbdgls = one, lnx1,lnx2 $
NAMELIST    ; hybrid = one, lnx1,lnx2,lnx1*lnx1, lnx2*lnx2 $
NAMELIST    ; translog = one, Log(x1), Log(x2), lnx1*lnx1, lnx2*lnx2, lnx1*lnx2 $
```

The hybrid and translog lists share two interactions, so these do not create separate table entries. You can see what is contained in the table with

```
LIST        ; [*] $
```

This is a special single purpose command that is used to obtain a listing of the internal table of interactions defined by your namelist commands. For the preceding, we would obtain

```
Constructed Variables Specified in Namelists and in Selection Equation
Variable  Variable   Variable ^ Power  Used by Namelist      Selection
----------------------------------------------------------------------
_ntrct01  LNX1      * LNX1             HYBRID   TRANSLOG
_ntrct02  LNX2      * LNX2             HYBRID   TRANSLOG
logX1          Log    X1               TRANSLOG
logX2          Log    X2               TRANSLOG
_ntrct05  LNX1      * LNX2             TRANSLOG
```

The internal names at the left are not necessarily meaningful. They will show up in some sets of results, however, so you may find the **LIST** command useful. When the variables needed to create an interaction term are deleted for any reason, then it is no longer possible to construct a namelist that uses that variable. You will receive a warning when this has occurred.

There is one table with up to 50 entries in it for namelists, so it is possible for you to overflow if you use too many interactions. As noted, it is possible to overflow the table if you have too many complicated namelists. You can clear the table with

```
DELETE       ; [*] $
```

This special command is used only to clear the namelists interaction terms definitions table. For our example, the **DELETE** command produces

```
Cleared internal table of constructed variables
Namelist HYBRID   is no longer defined.
Namelist TRANSLOG is no longer defined.
```

R8.4 Categorical Variables in Model Commands

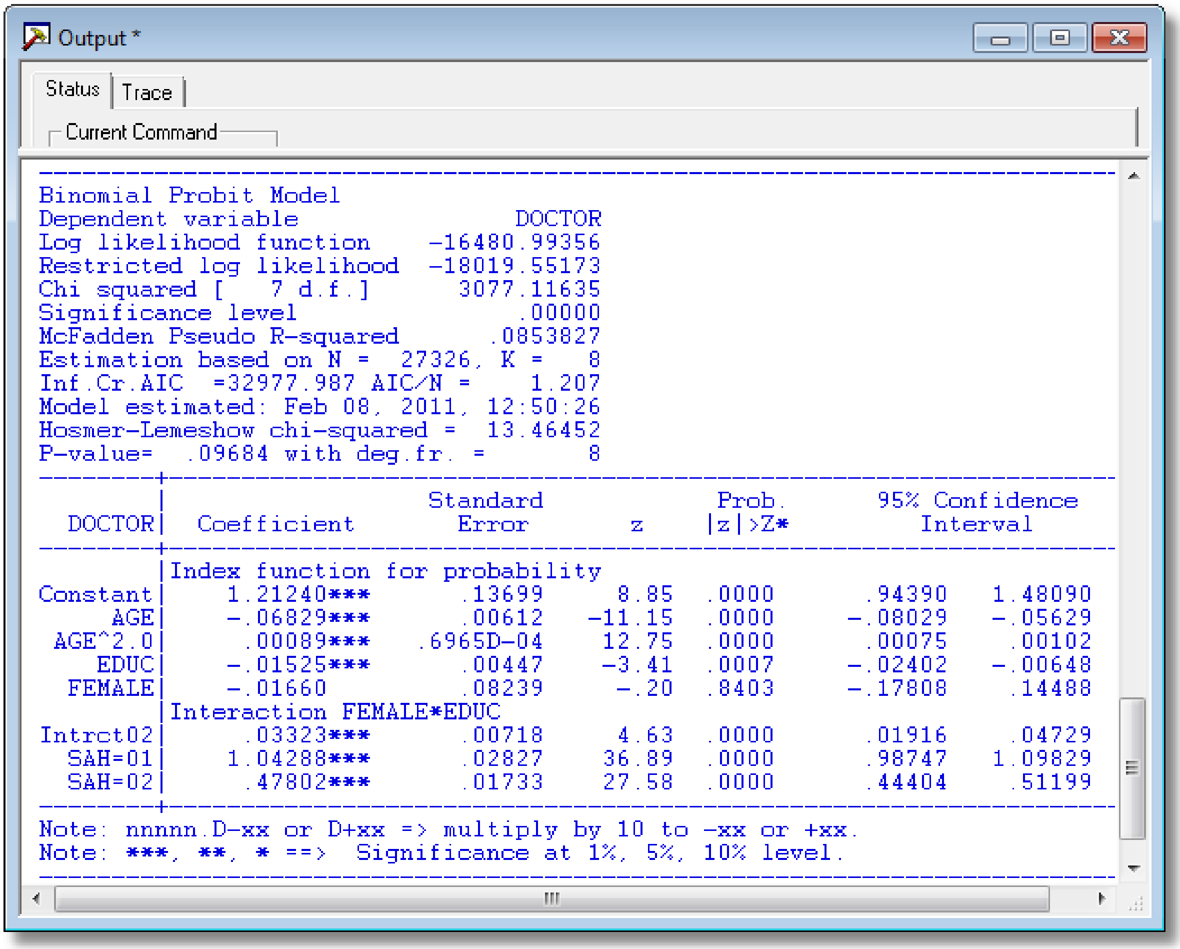
Categorical variables may be expanded in line in a model command with

Expand(variable) or #variable (such as Expand(ethnic) or #ethnic).

For example, the following commands first obtain the variable *sah* (self assessed health). *Hsat* is coded 0 – 10. The **RECODE** command collapses it to three categories. The **PROBIT** command then fits the model shown above with the expanded *sah* variable in the model. There are three categories, so one of them (the last one) is omitted.

```
CREATE       ; sah = hsat $
RECODE       ; sah ; 0/4 = 1 ; 5/7 = 2 ; 8/10 = 3 $
PROBIT       ; Lhs = doctor
             ; Rhs = one,age,age^2,educ,female,female*educ, #sah $
```

The resulting output is shown in Figure R8.3.



```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable                DOCTOR
Log likelihood function     -16480.99356
Restricted log likelihood   -18019.55173
Chi squared [    7 d.f.]      3077.11635
Significance level                .00000
McFadden Pseudo R-squared       .0853827
Estimation based on N =  27326, K =    8
Inf.Cr.AIC   =32977.987 AIC/N =    1.207
Model estimated: Feb 08, 2011, 12:50:26
Hosmer-Lemeshow chi-squared =  13.46452
P-value=   .09684 with deg.fr.  =       8
--------+--------------------------------------------------------------------
        |               Standard            Prob.       95% Confidence
  DOCTOR|  Coefficient    Error        z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|    1.21240***     .13699     8.85  .0000      .94390    1.48090
     AGE|    -.06829***     .00612   -11.15  .0000    -.08029    -.05629
 AGE^2.0|     .00089***   .6965D-04  12.75  .0000     .00075     .00102
    EDUC|    -.01525***     .00447    -3.41  .0007    -.02402    -.00648
  FEMALE|    -.01660        .08239    -.20   .8403    -.17808     .14488
        |Interaction FEMALE*EDUC
Intrct02|     .03323***     .00718     4.63  .0000     .01916     .04729
  SAH=01|    1.04288***     .02827    36.89  .0000     .98747    1.09829
  SAH=02|     .47802***     .01733    27.58  .0000     .44404     .51199
--------+--------------------------------------------------------------------
Note: nnnnn.D-xx or D+xx => multiply by 10 to -xx or +xx.
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Figure R8.3 Probit Model with Interaction Terms and Categorical Variable

The specification creates a temporary internal namelist, such as *educ* = xx with a set of up to 99 dummy variables of the form *educ* = 01, *educ* = nn… Your categorical variable need not be a sequence of integers, but it must be composed of integers that are somewhere in 1,...,100. Thus, *educ* could be coded 12,16,18,20 (number of years). The last dummy variable is always omitted, so this can create up to 99 dummy variables. They are temporary. The names will show up in the output by name, but will not show up in the data set or the project window. The variables and the temporary namelist vanish after the model is executed.

There is a special case, if the variable is named *year*, we can assume it is up to 100 years starting in 1921 and ending in 2020. Other restrictions:

1. This form may only be used for Rhs, Rh1, Rh2, Inst, Hfn, Hf1, Hfu, Hfv.
2. It may not be used in any form of multinomial choice model DISCRETE, NLOGIT, or any of the sub forms such as RPLOGIT, etc.
3. It may not be combined with the interaction terms in Section R8.3.1.

Note that the construction mimics **CREATE ; name = Expand(variable)** but does not put any new variables or namelists in the data set.

R8.5 Lags and Partial Differences in Model Commands

Models involving lagged variables may also be specified directly in terms of the lags, instead of using previously created variables. For example, a regression of y_t on *one* and y_{t-1} could be obtained with

```
CREATE      ; lagy = y[-1] $
REGRESS     ; Lhs = y ; Rhs = one, lagy $
```

But, these two lines could be replaced by the single command

```
REGRESS     ; Lhs = y ; Rhs = one, y[-1] $
```

Leads and lags are specified using *LIMDEP*'s usual format with square brackets. Leads can be specified with positive values in the brackets; the '+' is optional. I.e., **y[1]** and **y[+1]** are the same.

Lags and leads may be specified in this fashion in any Rhs, Rh1, Rh2, Inst, Eqn, or any other variable list in any model command. Note, however that an invalid attempt to use a lagged variable results in the diagnostic 'Variable list contains a name not in the expected table' followed by the offending name. For example,

```
LIST        ; x[-1] $
```

is invalid since **LIST** is not a model command.

TIP: Namelists may not contain logs or lags. These variables are computed 'on the fly,' and do not exist permanently in your data set unless you create them. If you attempt to include a lagged variable in a namelist, a diagnostic warning of an unidentified name, 'not in the expected table' will be given.

Logs and lagged variables can be mixed in any list of names, along with other variables, but not with each other. Thus, **Log(x[-1])** in a model command would be invalid. If a log cannot be computed, because of a nonpositive value, a -999 is returned, *BUT, NO WARNING IS ISSUED*. Lags or leads which extend beyond the limits of the data are returned as 0.0. You should set the sample carefully before you use either of these operations. By and large, *LIMDEP* is not able to account for your missing data here, *even if **SKIP** is turned on*.

If you are using an iterative procedure with a large data set, embedding lags and logs in the command will be a slower and much less efficient way to proceed, since the logs are recomputed during each pass through the data set. In this instance, you should compute the logs of the variables before calling for the procedure.

The following computes two step generalized least squares estimates for a classical regression with an AR(3) (third order autoregressive) disturbance. The **CREATE** command uses the coefficient vector that is automatically saved by the second regression. In the second regression, where residuals are regressed on three lagged values, the out of sample values are replaced with zeros.

```
SAMPLE      ; 1-127 $
CREATE      ; y = Rnn(0,1) ; x = Rnn(0,1) $
REGRESS     ; Lhs = y ; Rhs = one,x ; Res = e $
REGRESS     ; Lhs = e ; Rhs = e[-1], e[-2], e[-3],one $
CREATE      ; ygls = y - b(1)*y[-1] - b(2)*y[-2] - b(3)*y[-3]
            ; xgls = x - b(1)*x[-1] - b(2)*x[-2] - b(3)*x[-3] $
```

At this point, the first three rows of *xgls* and *ygls* are undefined.

```
SAMPLE      ; 4-127 $
REGRESS     ; Lhs = ygls ; Rhs = one,xgls $
```

A similar procedure could be used for other autoregressive schemes. But, for some applications, there is a simpler way to compute the last regression. The following option would normally be used mainly with the classical regression model but, in fact, can be used in any model command. If your model contains the specification

; Dfr = $r_1,r_2,...,r_p$

where there may be any number of coefficients, then every observation on every variable in your data set, $\mathbf{z}_t$ (except *one*, of course) is used *as if* it had been transformed to

$$\mathbf{z}_t^* = \mathbf{z}_t - r_1\mathbf{z}_{t-1} - \dots - r_p\mathbf{z}_{t-p}.$$

(The data are not actually transformed; observations are differenced as they are used.) For example, suppose you wish to regress $y_t^* = y_t - r_1 y_{t-1} - r_2 y_{t-2} - r_p y_{t-3}$ on the same transformation of a set of variables contained in a namelist, *x*. It is not necessary to compute the transformed variables. Use

```
REGRESS     ; Lhs = y ; Rhs = x ; Dfr = r1, r2, r3 $
```

Note that this could be applied to the example that embeds the lagged values in the regression. We could use

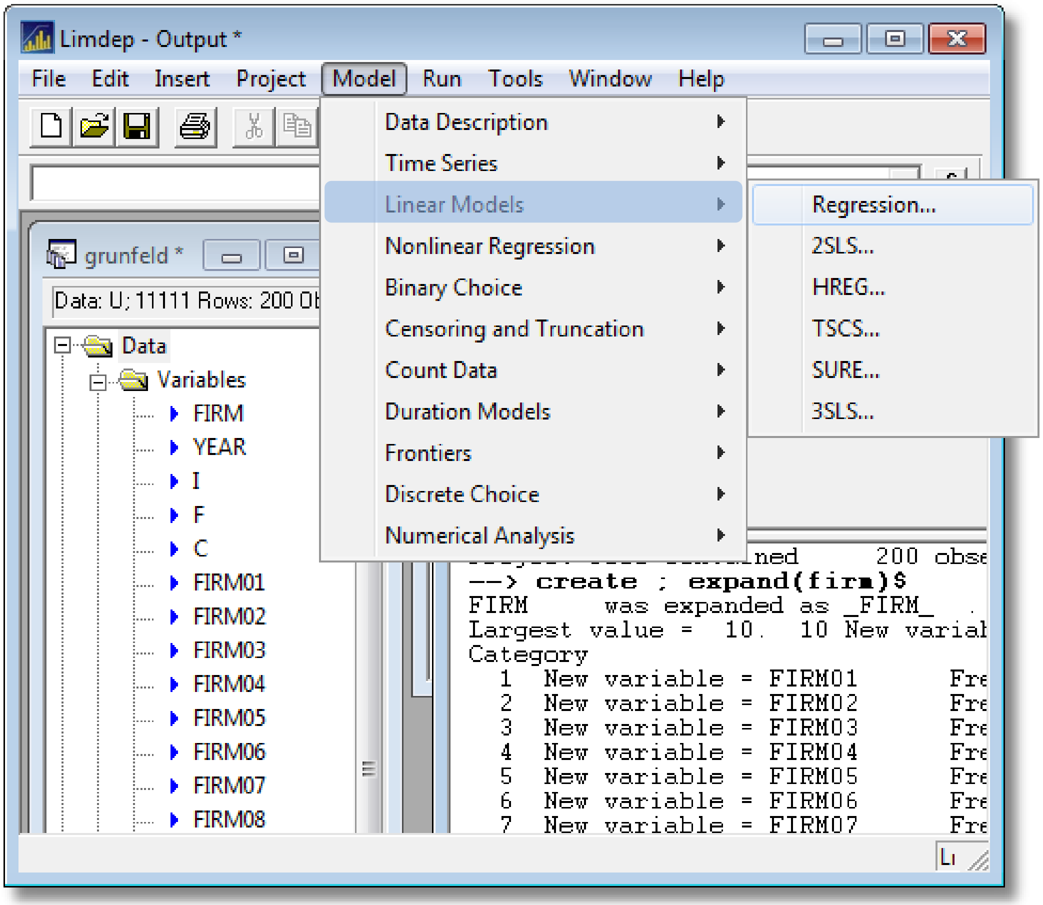
```
REGRESS     ; Lhs = y ; Rhs = one,x ; Dfr = b(1),b(2),b(3) $
```

and omit the preceding **CREATE** commands. The coefficients may take any value, including 1.0, so you can use this device to compute a regression in first differences;

```
REGRESS     ; Lhs = y ; Rhs = x ; Dfr = 1 $
```

R8.6 Command Builders

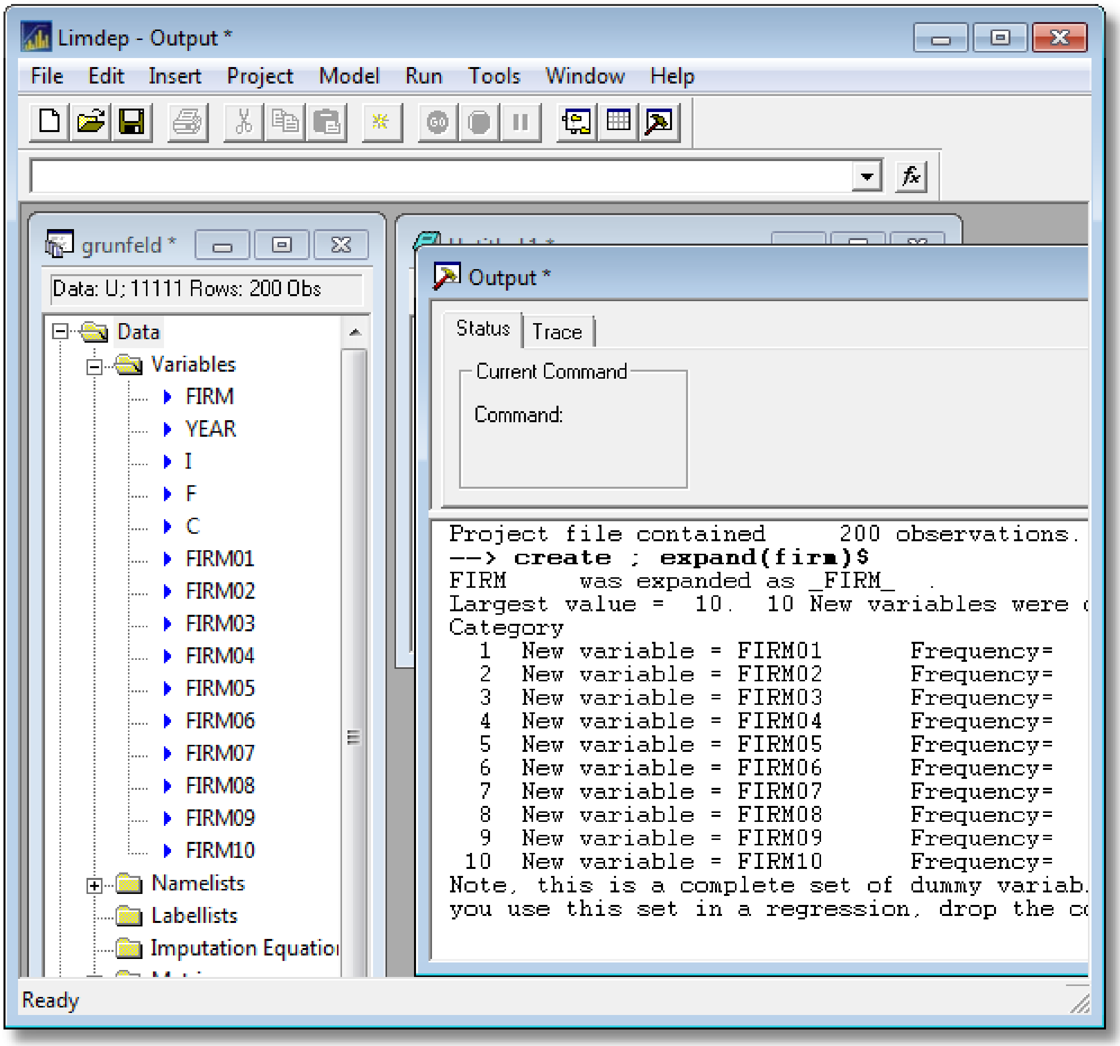
LIMDEP contains a set of dialog boxes and menus that you can use to build up your model commands in parts, as an alternative to laying out the model commands in the text editor. Figure R8.4 shows the top level model selection. The menu items, Data Description, etc., are subsets of the modeling frameworks that *LIMDEP* supports. We've selected Linear Models from the menu, which produces a submenu offering Regression, 2SLS, and so on. From here, the command builder contains specialized dialog boxes, specific to a particular model command.



```
                              ned      200 obse
--> create ; expand(firm)$
FIRM      was expanded as _FIRM_    .
Largest value =   10.   10 New varial
Category
   1   New variable = FIRM01       Fre
   2   New variable = FIRM02       Fre
   3   New variable = FIRM03       Fre
   4   New variable = FIRM04       Fre
   5   New variable = FIRM05       Fre
   6   New variable = FIRM06       Fre
   7   New variable = FIRM07       Fre
```

Figure R8.4 Model Command Builder with Linear Models Menu

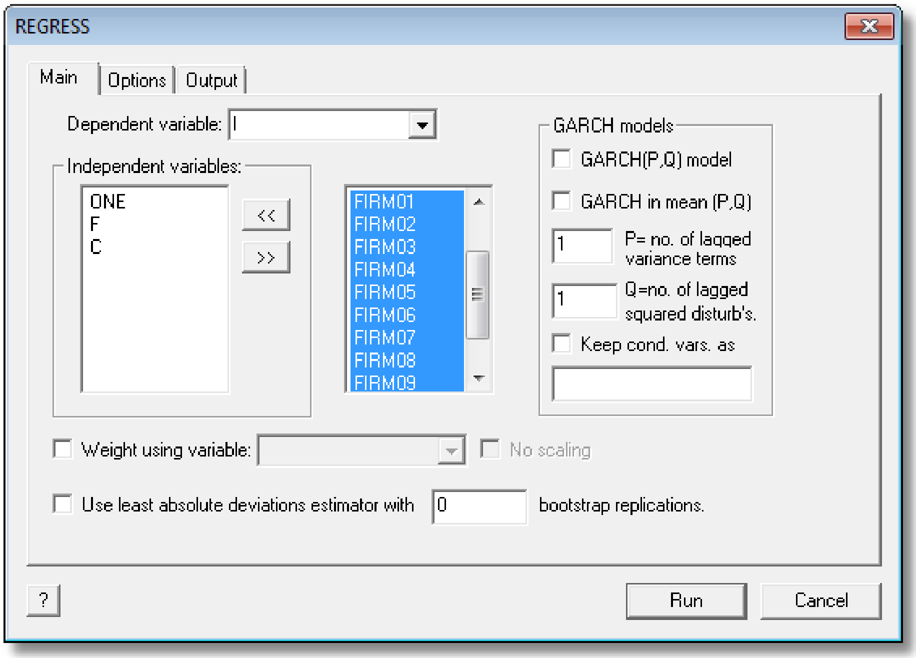
We'll illustrate operation of the command builder with a familiar application, Grunfeld's panel data set, 10 firms, 20 observations per firm, on three variables, investment, *i*, profit, *f*, and capital stock, *c*. The data are contained in the project shown in Figure R8.5. The variables, *firm*1, *form*2,… are dummy variables for the 10 firms. (The project files can be found in the resource folder created with installation: C:\LIMDEP11\Project Files.)



```
Project file contained      200 observations.
--> create ; expand(firm)$
FIRM      was expanded as _FIRM_  .
Largest value =  10.  10 New variables were c
Category
   1  New variable = FIRM01      Frequency=
   2  New variable = FIRM02      Frequency=
   3  New variable = FIRM03      Frequency=
   4  New variable = FIRM04      Frequency=
   5  New variable = FIRM05      Frequency=
   6  New variable = FIRM06      Frequency=
   7  New variable = FIRM07      Frequency=
   8  New variable = FIRM08      Frequency=
   9  New variable = FIRM09      Frequency=
  10  New variable = FIRM10      Frequency=
Note, this is a complete set of dummy variab.
you use this set in a regression, drop the co
```

Figure R8.5 Project Window for the Grunfeld Data

Figure R8.6 shows the main model specification dialog box (Main page), which will be quite similar for most of the models. The main window provides for specification of the dependent variable, the independent variables, and weights if desired. (Weights are discussed in Chapter R8.) We will not be using them in this example. If desired, the model is fully specified at this point. Note that the independent variables have been moved from window at the right, which is a menu, to the specification at the left. The highlighted variables D3 – D9 will be moved when we click the ‘<<’ button to select them. You may also click the query (?) button at the lower left of the dialog box to obtain a Help file description of the **REGRESS** command for linear models that is being assembled here. The Run button allows you now to submit the model command to the program to fit the model. There is also a box on the Main page for the **REGRESS** command for specifying the optional extension, the GARCH model. Since the main option box for this specification is not checked, this option will not be added to the model command.



Dependent variable: I

Independent variables:

ONE
F
C

FIRM01
FIRM02
FIRM03
FIRM04
FIRM05
FIRM06
FIRM07
FIRM08
FIRM09

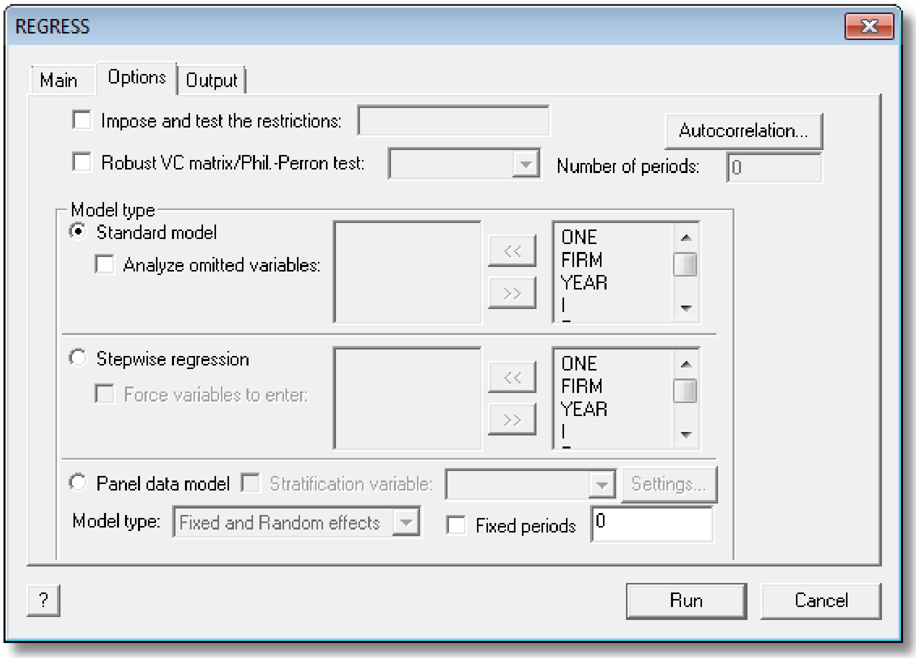
GARCH models
- ☐ GARCH(P,Q) model
- ☐ GARCH in mean (P,Q)
- 1 P= no. of lagged variance terms
- 1 Q=no. of lagged squared disturb's.
- ☐ Keep cond. vars. as

☐ Weight using variable: ☐ No scaling

☐ Use least absolute deviations estimator with 0 bootstrap replications.

Figure R8.6 Main Page of Command Builder for Linear Regression Model

The other two tabs in the command builder provide additional options for the linear regression model, as shown in Figures R8.7 and R8.8. For this example, we'll submit the simple command from the Main page with none of the options. Clicking the Run button submits the command to the program, and produces the output shown in Figure R8.9.



☐ Impose and test the restrictions:

☐ Robust VC matrix/Phil.-Perron test: Number of periods: 0

Model type
- ● Standard model
  - ☐ Analyze omitted variables:

ONE
FIRM
YEAR
I

- ○ Stepwise regression
  - ☐ Force variables to enter:

ONE
FIRM
YEAR
I

- ○ Panel data model ☐ Stratification variable:

Model type: Fixed and Random effects ☐ Fixed periods 0

Figure R8.7 Options Page for Linear Regression Model

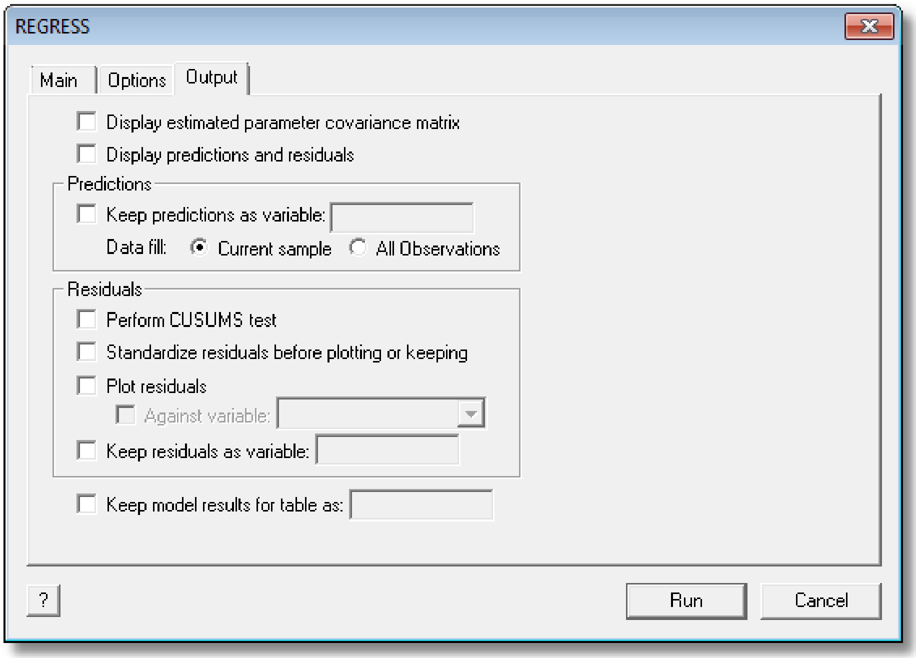
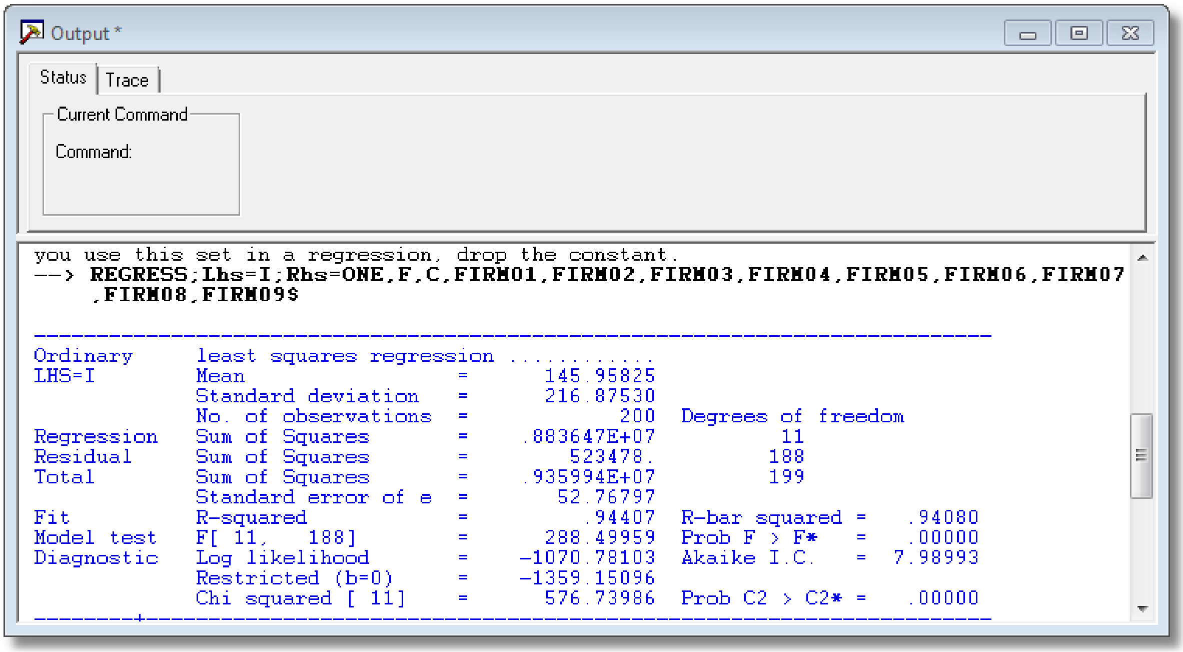

REGRESS
Main
Options
Output
Display estimated parameter covariance matrix
Display predictions and residuals
Predictions
Keep predictions as variable:
Data fill:
Current sample
All Observations
Residuals
Perform CUSUMS test
Standardize residuals before plotting or keeping
Plot residuals
Against variable:
Keep residuals as variable:
Keep model results for table as:
?
Run
Cancel

Figure R8.8 Output Page for Linear Regression Model

The regression command that was assembled by the command builder can be seen in the output window directly above the regression output in Figure R8.9:



```
you use this set in a regression, drop the constant.
--> REGRESS;Lhs=I;Rhs=ONE,F,C,FIRM01,FIRM02,FIRM03,FIRM04,FIRM05,FIRM06,FIRM07
    ,FIRM08,FIRM09$

-----------------------------------------------------------------------------
Ordinary     least squares regression ............
LHS=I        Mean                   =      145.95825
             Standard deviation     =      216.87530
             No. of observations    =            200  Degrees of freedom
Regression   Sum of Squares         =    .883647E+07          11
Residual     Sum of Squares         =        523478.         188
Total        Sum of Squares         =    .935994E+07         199
             Standard error of e    =       52.76797
Fit          R-squared              =         .94407  R-bar squared =    .94080
Model test   F[ 11,    188]         =      288.49959  Prob F > F*   =    .00000
Diagnostic   Log likelihood         =    -1070.78103  Akaike I.C.   =   7.98993
             Restricted (b=0)       =    -1359.15096
             Chi squared [ 11]      =      576.73986  Prob C2 > C2* =    .00000
---------+-------------------------------------------------------------------
```

Figure R8.9 Regression Results from Command Builder

The command builder has done the work of constructing the command and sending it to the program. Although the command builders do not remember their previous commands, the commands are available for you to reuse if you wish. You can use Edit:Copy and Edit:Paste to copy commands from your output window into your editing window, then just submit them from the editing window. The advantage of this is that you now need not reenter the dialog box to reuse the command. For example, if you wanted to add a time trend, *year*, to this equation, you could just copy the command to the editor, add *year* to the list, then select the line and click GO.

TIP: Commands that are 'echoed' to the output window are always marked with the leading '-- >.' The command reader will ignore these, so you can just copy and paste the whole line, or block of lines to move commands to your editing window.

The command builders are not complete for all models that can be specified by *LIMDEP*. Many features, such as the newer panel data estimators, are not contained in the command builders. In fact, you will probably 'graduate' from the dialog boxes fairly quickly to using the text editor for commands. The editors provide a faster and more flexible means of entering program instructions.

R8.7 Conditional Model Commands

There are several features available for conditioning model estimation for certain subsamples or for estimating models when some conditions are met, for example, based on some result from a previous model.

R8.7.1 Estimation Conditioned on a Scalar Test Value

Any model command may be conditional using

; (scalar = value) or # (# means not equal) or < or >

If the condition is met, everything continues. If it is not met, a diagnostic comes back and the model is not computed. The following shows an example for a regression model: The mean of a variable named *gcb* is computed. If this mean is greater than 200, the regression is computed. Since the mean is 110.9 which is less than the condition, the diagnostic is issued and the regression is not computed. Then, the same sequence is carried out with the condition that the mean is greater than 100. The mean passes this condition, and the regression is computed.

```
CALC        ; List ; gcb = Xbr(gc) $
            GCB = .11087976190476190D+03

REGRESS     ; (gcb> 200) ; Lhs = gc ; Rhs = one,invt,invc $
            Error 999: GCB is not > 200.00000. Model is not estimated.

REGRESS     ; (gcb> 100) ; Lhs = gc ; Rhs = one,invt,invc $
```

The **CALC** command obtains a value for *gcb* of 110.9. The first **REGRESS** command conditions on *gcb*> 200, which is not true, so the regression command is bypassed with a diagnostic. The second regression is computed, since 110.9 is greater than 100.

R8.7.2 Setting the Sample Temporarily for a Model

Any model instruction can be specified for a subset of the sample defined by any condition that can be used for an **INCLUDE** command. The syntax is

```
Model (any)    ; If [any condition that can be used for an INCLUDE command]
               ; ... the rest of model command $
```

The current sample is temporarily set to what is in the condition, relative to the current sample at the time. A standard case would be when one wishes to select on a binary variable, as in the following which computes separate regressions for men and women.

```
REGRESS        ; Lhs = wage ; Rhs = one,age,educ ; If [sex = 1] $
REGRESS        ; Lhs = wage ; Rhs = one,age,educ ; If [sex = 0] $
```

The command may be used more generally, as in the following in which the sample is set to include the observations for which a certain variable is less than the sample average.

```
CALC           ; gcb = Xbr(gc) $
REGRESS        ; If [gc<gcb] ; Lhs = gc ; Rhs = one,invc,invt $
```

Observation tags may be used as well. For example,

```
REGRESS        ; If [[ST_ABB]= AL] ; Lhs = hwy ; Rhs = one,pc $
```

R8.7.3 Looping over Strata for a Model Command

You may extend the **If [...]** feature above to request a model estimator to loop through a set of strata defined for a variable in the data set. The syntax is

```
Model (any)    ; For [variable] ; … the rest of the model $
```

This command executes once for each unique integer value of variable. To continue the earlier example,

```
REGRESS        ; For [firm] ; Lhs = i ; Rhs = one,f,c $
```

would fit a linear regression model for the subsamples firm = 1, firm = 2, and so on. This feature works for any set of integers – they need not be 1,2,... Data need not be sorted. The processor simply works through the data set and picks out the subsamples one at a time. You may narrow the definition with

```
Model (any)    ; For [variable = i1,i2,… list of integers] ; … $
```

as in

```
POISSON        ; For [educ = 9,12,16]
               ; Lhs = visits ; Rhs = one,age,income,educ $
```

(This feature corresponds to the 'by variable' types of construction in other commercial packages.)

To continue our earlier example, the Grunfeld data contain 20 years of data on each of 10 firms. The variable firm indexes the firms. To carry out the same regression for the 10 firms, we used

```
-->REGRESS ; For [firm] ; Lhs = i ; Rhs = one,f,c $
+--------------------------------------------------------+
| Setting up an iteration over the values of FIRM        |
| The model command will be executed for  10 values      |
| of this variable.  In the current sample of       200  |
| observations, the following counts were found:         |
| Subsample   Observations    Subsample  Observations    |
| FIRM     =   1        20    FIRM    =   2        20    |
| FIRM     =   3        20    FIRM    =   4        20    |
| FIRM     =   5        20    FIRM    =   6        20    |
| FIRM     =   7        20    FIRM    =   8        20    |
| FIRM     =   9        20    FIRM    =  10        20    |
+--------------------------------------------------------+
| Actual subsamples may be smaller if missing values     |
| are being bypassed.  Subsamples with 0 observations    |
| will be bypassed.                                      |
+--------------------------------------------------------+
*******************************************************************
*       Subsample analyzed for this command is FIRM     =       1 *
*******************************************************************
```

(The output includes individual estimation results for the 10 firms.)

The strata may be defined by the observation tags as well. For example, there are 11 regions defined in the data. The following uses three of them in a linear regression,

```
REGRESS    ; For [{REGION} = gf,mw,ma]
           ; Lhs = lgsp; Rhs = one,lpc,lhwy,lwater,lutil,lemp,ur $
```

```
----------------------------------------------------------
Setting up an iteration over the values of [REGION]
The model command will be executed for     3 values
of this variable.  In the current sample of     816
observations, the following counts were found:
Subsample  -Observations   Subsample  -Observations
[REGION] = GF         68   [REGION] = MW        119
[REGION] = MA        102
Subsamples within groups with GROUPSET = 1 are used
----------------------------------------------------
Actual subsamples may be smaller if missing values
are being bypassed.  Subsamples with 0 observations
will be bypassed.
----------------------------------------------------

-------------------------------------------------------------------------
Subsample analyzed for this command is [REGION] = GF       & GROUPSET = 1
```

(The output includes results for the three models.)

R8.8 Using Weights in Estimation

Any procedure which uses sums of the data, including descriptive statistics and all regression and nonlinear models can use a weighting variable by specifying

; Wts = name

where *name* is the name of the variable to be used for the weighting.

Any model based on least squares of any sort or on likelihood methods can be estimated with weights. This includes **REGRESS**, **PROBIT**, all **LOGIT** models, and so on. The only substantive exceptions are the nonparametric and semiparametric estimators, **MSCORE**, **NPREG**, and the Cox proportional hazard model.

NOTE: In computing weighted sums, the value of the variable, not its square root is used. As such, if you are using this option to compute weighted least squares for a heteroscedastic regression, *name* should contain the reciprocals of the disturbance variances, not the standard deviations.

In maximum likelihood estimation, the terms in the log likelihood and its derivatives, not the data themselves, are multiplied by the weighting variable. That is, when you provide a weighting variable, *LIMDEP* computes a sum of squares and cross products in a matrix as $\mathbf{X'WX} = \Sigma_i w_i \mathbf{x}_i \mathbf{x}'$ and a log likelihood Log $L = \Sigma_i w_i \log(f_i)$, where w_i is an observation on your weighting variable.

The weighting variable must always be positive. The variable is examined before the estimation is attempted. If any nonpositive values are found, the estimation is aborted.

During computation, weights are automatically scaled so that they sum to the current sample size. The variable, itself, is not changed, however. If you specify that variable w is to be the weighting variable in **; Wts = w**, the weight actually applied is $w_i{}^* = [N/\Sigma_i w_i] \times w_i$. This scaling may or may not be right for a selected sample in a sample selection model. That is, after selection, the weights on the selected data points may or may not sum to the number of selected data points. As such, the weights in **SELECT** with univariate and bivariate probit criterion equations are rescaled so that they sum exactly to the number of selected observations.

TIP: The scaling will generally not affect coefficient estimates. But, it will affect estimated standard errors, sometimes drastically.

To suppress the scaling, for example for a grouped data set in which the weight is a replication factor, use **; Wts = name,noscale** or just **; Wts = name,n**.

WARNING: When this option is used with grouped data qualitative choice models, such as logit, it often has the effect of enormously reducing standard errors and blowing up t-ratios.

The ‘noscale’ option would most likely be useful when examining proportions data with a known group size. For example, consider a probit analysis of county voting returns. The data would consist of N observations on $[n_i, p_i, x_i]$, where n_i is the county size, p_i is the proportion of the county population voting on the issue under study, and x_i is the vector of covariates. Such data are heteroscedastic, with the variance of the measured proportion being proportional to $1/n_i$. We emphasize, once again, when using this option with population data, standard errors tend to become vanishingly small, and call upon the analyst to add the additional measure of interpretation.

R9: Output

R9.1 Standard Output from Estimation Programs

Results produced by an estimation commands will vary from model to model. The display below that would appear in the output window would be typical. These are the results produced by estimation of a basic tobit model using the Mroz.lpj data provided with the program. (The project files can be found in the resource folder created with installation: C:\LIMDEP11\Project Files.) The display begins with an echo of the estimation command followed by a statement of the exit status. Status=0 means that the model was successfully estimated. (Something other than zero will indicate a problem and will be accompanied by a diagnostic message. The sign on the log likelihood in the status line is reversed because the optimizer minimizes the negative of the log likelihood.)

```
-->TOBIT ; Lhs = whrs ; Rhs = one,wa,we,ww,kl6,k618 $
Normal exit:   5 iterations. Status=0, F=    3817.300
-----------------------------------------------------------------------------
Limited Dependent Variable Model - CENSORED
Dependent variable                 WHRS
Log likelihood function     -3817.29976
Estimation based on N =    753, K =   7
Inf.Cr.AIC  = 7648.600 AIC/N =   10.158
Threshold values for the model:
Lower=     .0000     Upper=+infinity
LM test [df] for tobit=    316.766[  6]
Normality Test, LM    =     40.989[  2]
ANOVA  based fit measure =    .225490
DECOMP based fit measure =    .236441
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
    WHRS|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Primary Index Equation for Model
Constant|    1702.85***    444.9422     3.83  .0001      830.78   2574.93
      WA|   -33.9421***    7.09484     -4.78  .0000    -47.8477  -20.0364
      WE|   -9.39496       22.09910     -.43  .6707   -52.70840  33.91847
      WW|    199.156***    15.52898    12.82  .0000     168.719   229.592
     KL6|   -837.633***    118.4994    -7.07  .0000   -1069.888  -605.379
    K618|   -103.922***    39.42369    -2.64  .0084    -181.191   -26.653
        |Disturbance standard deviation
   Sigma|    1158.25***    42.96400    26.96  .0000     1074.04   1242.46
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
```

Most of the models estimated by *LIMDEP* (including the tobit model above) are single equation, ‘index function’ models. There is a dependent variable, which we’ll denote ‘*y*,’ a set of independent variables, ‘*x*,’ and a model, consisting, in most cases, of either some sort of regression equation or a statement of a probability distribution, either of which depends on an index function, $\mathbf{x}'\boldsymbol{\beta}$ and a set of ‘ancillary’ parameters, $\boldsymbol{\theta}$, such as a variance term, σ^2 in a regression or a tobit model. The parameters to be estimated are $[\boldsymbol{\beta},\boldsymbol{\theta}]$.

Some notes about the output:

- Results always include the coefficients, standard errors, and ratios of coefficients to standard errors. In the index function models, the coefficients are named by the variable that multiplies them in the index function. In models which do not use an index function (**NLSQ**, **MINIMIZE**, **CLOGIT**, and a several others), the parameter label that you provide will appear with the estimate instead. Note that 'one' becomes 'Constant' in the table.

- The prob value shown, 'Prob[|*z*| >*z**],' is the value for a two tailed test of the hypothesis that the coefficient equals zero. The probability shown is based on the standard normal distribution in all cases except the linear regression model, when it is based on the 't' distribution with degrees of freedom that will be shown in the table. When you fit a linear regression, the table will list values of 't' and Prob[|*t*|>*t**]

- The diagnostics table for the model reports some statistics which will be present for all models usually including:

  1. left hand side variable
  2. number of observations used and the number of parameters estimated
  3. the date and time the model was estimated
  4. log likelihood function or other estimation criterion function

- Some results will be computable only for some models. The following results listed for the Poisson model will not appear when there is no natural, nested hypothesis to test. (For example, they will not normally appear in the output for the tobit model.)

  1. log likelihood at a restricted parameter estimate, usually zero
  2. chi squared test of the restriction,
  3. significance level
  4. degrees of freedom

- Finally, there are usually some statistics or descriptors which apply specifically to the model being estimated. For the tobit model, the output contains the threshold values used for the censoring. There are also two Lagrange multiplier based specification test statistics that are specific to the tobit model.

- Footnotes to the table will explain specific features of the output. This will usually include a legend about reports of statistical significance, such as in the previous example. But, other information might be included as well. For example, the result below shows the standard display of partial effects for a probit model based on our earlier example. (The Lhs variable in the tobit equation is hours worked. In the probit model that produced the results below, it is 1[hours > 0].

The footnotes for this table indicate how the elasticity is computed for a binary variable and notes that the '#' indicates a dummy variable in the model, for which the partial effect is computed by first difference rather than differentiation.

```
-----------------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with
respect to the vector of characteristics
Average partial effects for sample obs.
--------+--------------------------------------------------------------------
        |     Partial                            Prob.      95% Confidence
     LFP|      Effect    Elasticity      z     |z|>Z*          Interval
--------+--------------------------------------------------------------------
      WA|    -.00724***      -.54202    -2.78  .0055      -.01235   -.00213
      WE|     .03956***       .85514     5.17  .0000       .02455    .05457
    KIDS|    -.11704***      -.14330    -2.65  .0080      -.20355   -.03053   #
--------+--------------------------------------------------------------------
#  Partial effect for dummy variable is E[y|x,d=1] - E[y|x,d=0]
Elasticity for a binary variable is marginal effect/Mean.
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

R9.1.1 Changing the Confidence Level for the Confidence Intervals

The default level of confidence for the confidence intervals is the universal standard 95%. You can change this by adding

; Clevel = the value.

Acceptable values are from .10 to .99. For example, by adding **; Clevel = .90** to the **TOBIT** command in the first example above, we revise the results to obtain the following output. The results are the same as before save for the narrower (now 90% level) confidence intervals.

```
--------+--------------------------------------------------------------------
        |                  Standard              Prob.      90% Confidence
    WHRS|  Coefficient       Error         z     |z|>Z*          Interval
--------+--------------------------------------------------------------------
        |Primary Index Equation for Model
Constant|    1702.85***    444.9422      3.83  .0001       970.99   2434.72
      WA|   -33.9421***     7.09484     -4.78  .0000     -45.6120  -22.2721
      WE|   -9.39496       22.09910      -.43  .6707    -45.74474  26.95481
      WW|    199.156***    15.52898     12.82  .0000      173.613   224.699
     KL6|   -837.633***    118.4994     -7.07  .0000    -1032.547  -642.719
    K618|   -103.922***    39.42369     -2.64  .0084     -168.769   -39.076
        |Disturbance standard deviation
   Sigma|    1158.25***    42.96400     26.96  .0000      1087.58   1228.92
--------+--------------------------------------------------------------------
```

The setting for the confidence level is temporary, only for that model, when it is embedded in a model command. You can make the change permanent with the full command

DEFAULT ; Clevel = the value $

The setting will be the new default for all models from that point on.

R9.1.2 Information Criteria for Maximum Likelihood Estimators

As seen in the first set of model results, *LIMDEP* reports the Akaike Information Criterion, or AIC with all maximum likelihood estimates. For the example,

```
Inf.Cr.AIC  = 7648.600 AIC/N =   10.158
```

AIC is computed as -2LnL + 2K where lnL is the log likelihood function and K is the number of parameters. AIC is similar to adjusted R^2 in regression, but in general, a lower AIC is better. Some AIC grows with the sample size, N. *LIMDEP* also reports AIC/N, for convenience.

Other authors have suggested similar measures with different corrections for the model size. A ‘finite sample’ version of AIC that may have better small sample properties is

$$FSAIC \ = \ \text{AIC} + \ 2\frac{K(K+1)}{N-K-1}.$$

The Bayes Information Criterion is

$$BIC \quad = \ -2\ln L + K\ln N$$

While the Hannan and Quinn Information Criterion is

$$HQIC \ = -2\ln L + 2K \text{ Ln Ln}N$$

You can request to display these additional measures by adding

; Output = IC

to your model command. For our example, the single line of information criterion results will be replaced by

```
Inf.Cr.AIC  = 7648.600 AIC/N =   10.158
FinSmplAIC  = 7648.750 FIC/N =   10.158
Bayes IC    = 7680.968 BIC/N =   10.200
HannanQuinn = 7661.069 HIC/N =   10.174
```

The switch remains on until you turn it off with **; Output = NoIC**.

R9.1.3 Timing Model Estimation

You can display the time required to estimated your models with the command

TIMER $

This is a switch that will remain on until you turn it off with

NOTIMER $

For example, if we issued a **TIMER** command before our **TOBIT** command in the first example above, the additional line

```
Elapsed time:     0 hours,  0 minutes,   .06 seconds.
```

will appear after the results. (Note that this result might differ from one computer to another for identical models using the same data set.)

The timer will usually just help you see how very fast modern computers are. However, there is one use for the execution timer that is likely to be very useful. *LIMDEP* contains many simulation based estimators that do require a very large amount of time. Using **TIMER** with a small pilot execution can help in planning for estimation of a full model. To continue the earlier example, we fit a random parameters tobit model with a random constant term and random coefficient on the wage variable. We used 100 Halton draws for the simulations. The command is

```
TOBIT         ; Lhs = whrs ; Rhs = one,wa,we,ww,kl6,k618
              ; Rpm ; Fcn = one(n),ww(n)
              ; Pts = 100 ; Halton $
```

After the results return, we are informed

```
Elapsed time:     0 hours,  0 minutes, 27.27 seconds.
```

The data set used in this example is one third of the full data set. When we will fit the full specification of this model using the entire data set, and allowing all six coefficients to be random, we will use 1,000 Halton draws rather than 100. How long will it take? The time needed for the simulation based estimator is roughly linear in the number of draws, the number of observations and the number of random parameters. Based on the preceding, the estimated time it will take is 10×3×3×27.27 seconds, or about 41.4 minutes. If you are doing this style of estimation with very large data sets, it can be useful to plan on how long the estimation will take.

R9.2 Initial Model Results

Nearly all models fit by *LIMDEP* are nonlinear and estimation requires an iterative optimization. Starting values for the iterations are usually obtained by estimating a simpler model, often using ordinary least squares, but sometimes by using maximum likelihood or some other technique.

R9.2.1 Displaying Initial Least Squares Estimates

Ordinary least squares (OLS) will frequently be used to obtaining the default starting values for the iterations. However, the OLS estimator is occasionally an interesting entity in its own right. To see the initial OLS outputs when they are computed for a nonlinear model, add

; OLS

to your model command.

NOTE: In order to reduce the amount of superfluous output, OLS results are not reported automatically except for the linear regression model.

```
-----------------------------------------------------------------------------
Ordinary    least squares regression ............
LHS=WHRS    Mean                 =      740.57636
            Standard deviation   =      871.31422
            No. of observations  =            753  Degrees of freedom
Regression  Sum of Squares       =    .131978E+09          5
Residual    Sum of Squares       =    .438932E+09        747
Total       Sum of Squares       =    .570910E+09        752
            Standard error of e  =      766.54595
Fit         R-squared            =         .23117  R-bar squared =   .22603
Model test  F[  5,   747]        =       44.92158  Prob F > F*   =   .00000
Diagnostic  Log likelihood       =    -6066.79538  Akaike I.C.   = 13.29173
            Restricted (b=0)     =    -6165.77240
            Chi squared [  5]    =      197.95404  Prob C2 > C2* =   .00000
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
    WHRS|  Coefficient     Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|   1567.05***    267.1143     5.87  .0000     1043.51   2090.58
      WA|  -18.3965***     4.21727    -4.36  .0000    -26.6622  -10.1308
      WE|  -7.49913       13.16210    -.57  .5688   -33.29638  18.29811
      WW|   104.457***     9.26340    11.28  .0000      86.301   122.612
     KL6|  -392.617***    60.66255    -6.47  .0000    -511.513  -273.721
    K618|  -78.6937***    23.23463    -3.39  .0007   -124.2327  -33.1546
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

TIP: The OLS estimator is almost never a consistent estimator of the parameters of the nonlinear models estimated by *LIMDEP*.

R9.2.2 Intermediate Model Estimates

Occasionally, obtaining estimates of the parameters of a nonlinear model begins with estimation of a restricted version of that model. For examples: Estimation of a negative binomial model begins with estimation of a Poisson regression model, again to obtain the starting values. Estimation of the linear sample selection model by MLE begins with Heckman's two step least squares estimator. In cases like these, your results will often contain full sets of output for both the initial, restricted model and the final model that you specified in your command.

The following results are based on the ship accident data used in Greene (2012), Table F18.3. (The data are ship-accidents.lpj in the program provided data sets.) (Some of the model results are not reported.) The intermediate Poisson results might be useful, however, the second set of results are the main results for the command. (The OLS estimator was also computed before the Poisson estimator, but not reported.)

```
CREATE      ; logmth = Log(months) $
NAMELIST    ; x = one,ta,tc,td,te,t6064,t6569,t7074,o7579 $
SKIP $
NEGBIN      ; Lhs = num ; Rhs = x,logmth $
```

```
---------------------------------------------------------------
Deleted      6 observations with missing data. N is now     34
---------------------------------------------------------------
-----------------------------------------------------------------------------
Poisson Regression
Dependent variable                  ACC
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
     ACC|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|   -5.61566***      .98586    -5.70  .0000    -7.54791   -3.68341
  ...
  LOGMTH|     .90617***      .10175     8.91  .0000      .70675    1.10559
--------+--------------------------------------------------------------------
Line search at iteration 43 does not improve fn. Exiting optimization.
-----------------------------------------------------------------------------
Negative Binomial Regression
Dependent variable                  ACC
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
     ACC|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|   -5.61780***     1.24854    -4.50  .0000    -8.06489   -3.17070
  ...
  LOGMTH|     .90633***      .12617     7.18  .0000      .65903    1.15362
        |Dispersion parameter for count data model
   Alpha| .44618D-04         .02567      .00  .9986 -.50276D-01  .50365D-01
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

R9.3 Using DISPLAY to View Estimation Results

When your own program computes an estimate of a parameter vector and an asymptotic covariance matrix, you can use **DISPLAY** to show the results in the standard format. The general form of the instruction is

```
DISPLAY     ; Parameters = the vector of estimated parameters
            ; Covariance = the estimated covariance matrix $
```

The instruction constructs a table of results with standard errors, 'z' ratios and confidence intervals in the same form of the standard output. For example,

```
TOBIT       ; Lhs = y ; Rhs = x $
DISPLAY     ; Parameters = b ; Covariance = varb $
```

would display the set of estimates for the tobit model twice. There are four optional specifications:

; Labels = appropriate list of labels
; Title = a title to use in the results
; Logl = a log likelihood to be displayed with the results
; Test: hypothesis tests (This feature is discussed in Chapter R13.)

The command that appears at the end of the SCLS program,

```
DISPLAY     ; Labels = x
            ; Parameters = bj
            ; Covariance = v
            ; Title = Symmetric Censored Least Squares $
```

produces the results below.

```
-----------------------------------------------------------------------------
Symmetrically Censored Least Squares
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
      YS|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    2076.64***     492.0428     4.22  .0000     1112.25    3041.02
      WA|   -31.0198***      8.74809    -3.55  .0004    -48.1657   -13.8738
      WE|   -4.53505        28.42455     -.16  .8732   -60.24615   51.17606
      WW|    101.108***     34.08620     2.97  .0030      34.300    167.915
     KL6|   -869.048***     191.9083    -4.53  .0000   -1245.181   -492.914
    K618|   -135.082***     46.39106    -2.91  .0036    -226.007    -44.157
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Note the use of the namelist to provide the labels. Namelists are discussed in Chapter R6.

R9.4 Covariance Matrices, Predictions and Hypothesis Tests

There are several additional sets of results that can be reported with the estimation output, including plots, lists of fitted values, hypothesis tests, and so on.

R9.4.1 Displaying Covariance Matrices

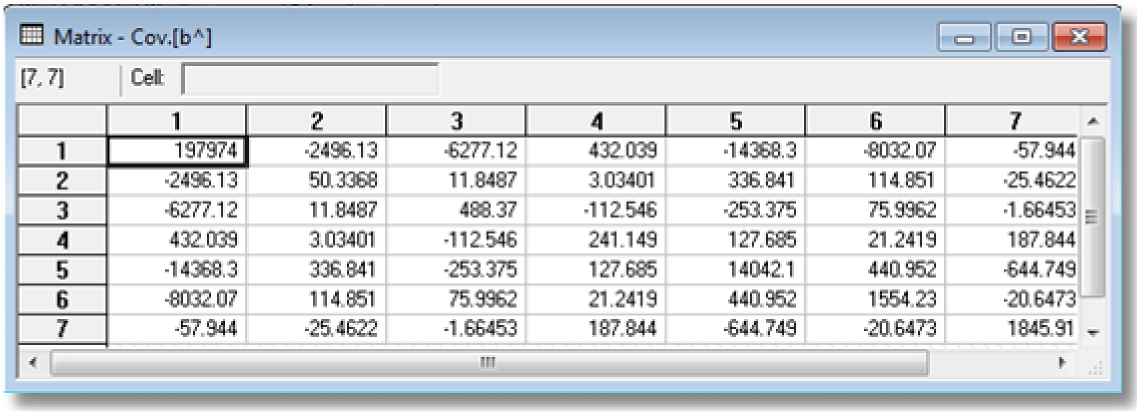
The output display generally does not is the estimate of the asymptotic covariance matrix of the estimates. Since models can have up to 150 parameters, this part of the output is potentially voluminous. Consequently, the default is to omit it. You can request that it be listed by adding

; Covariance Matrix

to the model command. (Previous versions of *LIMDEP* and *NLOGIT* used **; Printvc** for this switch. This syntax is still supported.) Since covariance matrices can be extremely large, this is handled two ways. If the resultant matrix is 5×5 or smaller, it is included in the output listing. The earlier tobit equation had six independent variables plus the estimate of σ. If we remove the last two variables from the namelist, the displayed results are as follows. (Some of the results are omitted.)

```
-->NAMELIST ; x = one,wa,we,ww $
-->TOBIT    ; Lhs = y ; Rhs = x ; Covariance Matrix $
Normal exit:   5 iterations. Status=0, F=    3846.188
-----------------------------------------------------------------------------
Limited Dependent Variable Model - CENSORED
Dependent variable                    Y
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
       Y|  Coefficient      Error       z    |z|>Z*        Interval
--------+--------------------------------------------------------------------
        |Primary Index Equation for Model
Constant|    370.181       397.0084      .93  .3511    -407.942  1148.303
      WA|   -7.82602        6.15057    -1.27  .2032   -19.88091   4.22887
      WE|   -22.7934       22.69898    -1.00  .3153    -67.2826   21.6958
      WW|    221.558***    16.14246    13.73  .0000     189.920   253.197
        |Disturbance standard deviation
   Sigma|    1212.37***    45.20887    26.82  .0000     1123.76   1300.98
--------+-----------------------------------------------------------------------
Cov.[b^]|          ONE             WA             WE             WW          Sigma
--------+-----------------------------------------------------------------------
ONE|       157616.       -1767.45       -6642.98       749.481       -801.367
      WA|      -1767.45        37.8295        14.4632       -2.85004       -11.2469
      WE|      -6642.98        14.4632        515.244       -118.841       -16.7283
      WW|       749.481       -2.85004       -118.841        260.579        220.413
   Sigma|      -801.367       -11.2469       -16.7283        220.413        2043.84
```

If the matrix has more than five columns, then it is offered as an additional matrix in the project window with the output, as shown in Figure R9.1 for a larger tobit model. When estimation is done in stages, **; Covariance** will only produce an estimated covariance matrix at the final step. Thus, no covariance matrix is displayed for initial least squares results.


Matrix - Cov.[b^]

[7, 7] Cell:

	1	2	3	4	5	6	7
1	197974	-2496.13	-6277.12	432.039	-14368.3	-8032.07	-57.944
2	-2496.13	50.3368	11.8487	3.03401	336.841	114.851	-25.4622
3	-6277.12	11.8487	488.37	-112.546	-253.375	75.9962	-1.66453
4	432.039	3.03401	-112.546	241.149	127.685	21.2419	187.844
5	-14368.3	336.841	-253.375	127.685	14042.1	440.952	-644.749
6	-8032.07	114.851	75.9962	21.2419	440.952	1554.23	-20.6473
7	-57.944	-25.4622	-1.66453	187.844	-644.749	-20.6473	1845.91

Figure R9.1 Covariance Matrix after Estimation

R9.4.2 Listing and Saving Model Predictions and Residuals

Most estimated models produce several results based on the results of estimation. Predictions from estimated models are saved in the data set as new variables and/or listed with the output. In some cases, residuals are also computed. The additional variables will vary from one model to the next. In some cases, such as ordered probit models, neither fitted values nor residuals are meaningful. In other cases, such as the binary probit model, there is a meaningful model prediction, but ‘residuals’ are not meaningful. To obtain a listing of model predictions with the estimation results, add

; List

to the model command.

The following are added to the model results for the tobit model fit earlier when the command contains **; List**.

```
Predicted Values          (* => observation was not in estimating sample.)
Observation          Observed Y   Predicted Y   Residual        x(i)b    Pr[Nonlim]
        1            1610.0000    648.34116    961.65884    334.30377    .6135671
        2            1656.0000    851.31070    804.68930    640.61548    .7098988
        3            1980.0000    545.36992    1434.6301    158.00438    .5542540
        4            456.00000    653.49932   -197.49932    342.69169    .6163354
        5            1568.0000    681.84104    886.15896    388.12603    .6312240
        6            2032.0000    895.19098    1136.8090    701.65938    .7276738
        7            1440.0000    1781.6688   -341.66884    1748.4578    .9344232
        8            1020.0000    1392.7813   -372.78129    1319.2411    .8726474
        9            1458.0000    555.67710    902.32290    176.49569    .5605569
       10            1600.0000    1117.9592    482.04082    992.07176    .8041467
```

WARNING: You might prefer not to use this feature if you have a very large sample.

The observed data are shown as they were used in estimation. The prediction will vary from model to model. When shown, a prediction is generally the conditional mean function. But, each model described in the *Econometric Modeling Guide* will include details about the form of the conditional mean function. Note, for example, in the tobit model, although it is a single index model, the prediction is not equal to the index function, as can be seen above. Listings such as this one will usually also contain a variable that is specific to the model being estimated. The listing for the tobit model shows the index function and the estimated probability that the dependent variable is positive. If the model were fit as a linear regression model, then the listing would appear as

```
Predicted Values          (* => observation was not in estimating sample.)
Observation        Observed Y   Predicted Y   Residual     95% Forecast Interval
        1          1610.0000    846.10168    763.89832   -662.93686    2355.1402
        2          1656.0000    912.85679    743.14321   -594.93618    2420.6498
        3          1980.0000    679.29119    1300.7088   -828.64375    2187.2261
        4          456.00000    730.03419   -274.03419   -776.47476    2236.5431
        5          1568.0000    821.40877    746.59123   -685.62023    2328.4378
        6          2032.0000    978.99270    1053.0073   -527.41754    2485.4029
        7          1440.0000    1479.4728   -39.472813   -29.413349    2988.3590
        8          1020.0000    1302.9123   -282.91233   -206.24767    2812.0723
        9          1458.0000    658.73631    799.26369   -845.82208    2163.2947
       10          1600.0000    1091.8489    508.15111   -412.84736    2596.5452
```

The last two columns now contain a 95% forecast interval based on the linear regression model.

To retain the model predictions as a new variable in the data set, include

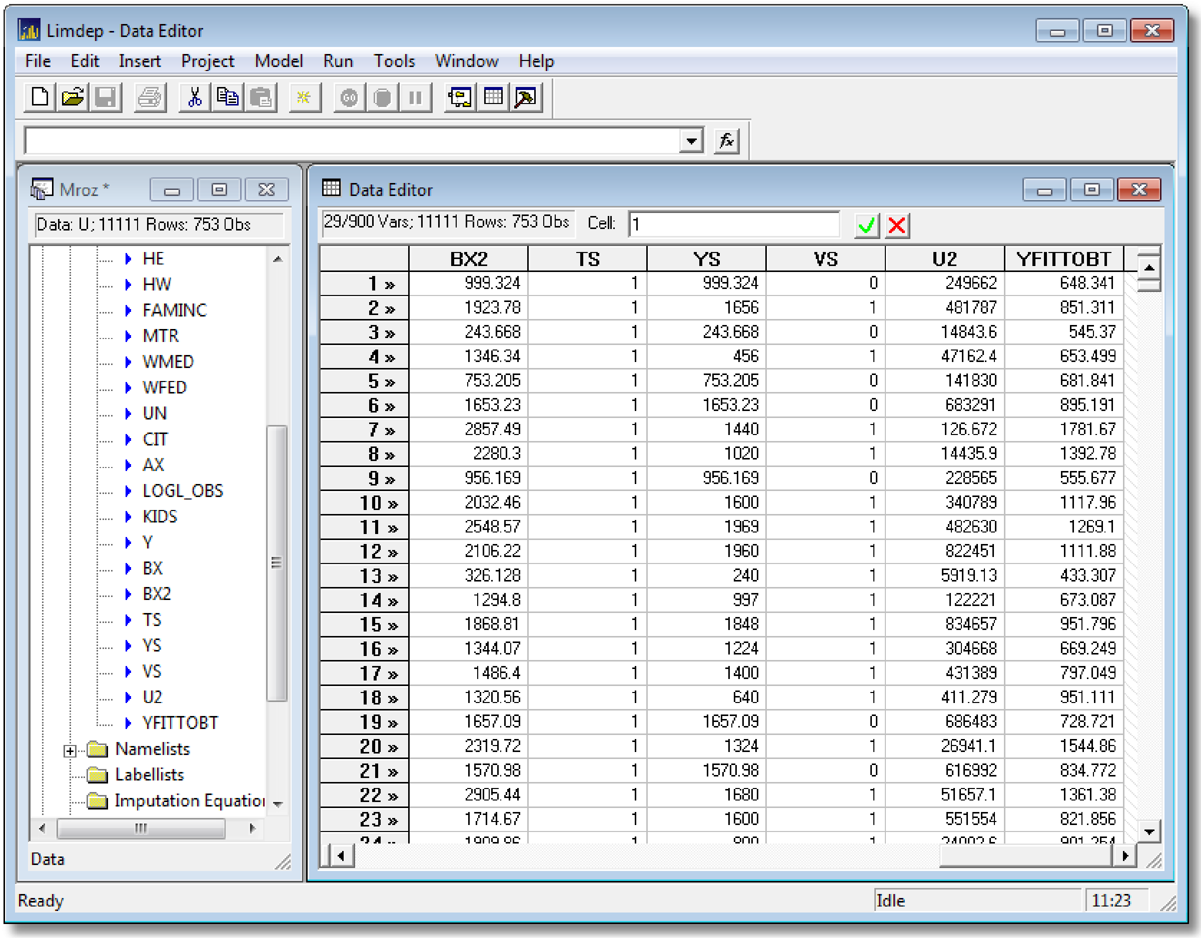
; Keep = the name for the new variable.

Continuing our tobit example, we added **; Keep = yfittobt** to the command. Figure R9.2 shows the changed project window and the new variable in the data area.

This feature computes predictions for the observations in the current sample. (The current sample is described in Chapter R7.) If this is not the full sample, then observations in the data set that were not used in estimation are left as missing values, -999. You can use the model to fill these missing values by adding

; Fill

to the command. *LIMDEP* will use the model to predict as many of these observations as possible. If there are missing values among the independent variables, then the observation will be left as missing. Missing values of the dependent variable do not prevent filling the observations, however.



	BX2	TS	YS	VS	U2	YFITTOBT
1 »	999.324	1	999.324	0	249662	648.341
2 »	1923.78	1	1656	1	481787	851.311
3 »	243.668	1	243.668	0	14843.6	545.37
4 »	1346.34	1	456	1	47162.4	653.499
5 »	753.205	1	753.205	0	141830	681.841
6 »	1653.23	1	1653.23	0	683291	895.191
7 »	2857.49	1	1440	1	126.672	1781.67
8 »	2280.3	1	1020	1	14435.9	1392.78
9 »	956.169	1	956.169	0	228565	555.677
10 »	2032.46	1	1600	1	340789	1117.96
11 »	2548.57	1	1969	1	482630	1269.1
12 »	2106.22	1	1960	1	822451	1111.88
13 »	326.128	1	240	1	5919.13	433.307
14 »	1294.8	1	997	1	122221	673.087
15 »	1868.81	1	1848	1	834657	951.796
16 »	1344.07	1	1224	1	304668	669.249
17 »	1486.4	1	1400	1	431389	797.049
18 »	1320.56	1	640	1	411.279	951.111
19 »	1657.09	1	1657.09	0	686483	728.721
20 »	2319.72	1	1324	1	26941.1	1544.86
21 »	1570.98	1	1570.98	0	616992	834.772
22 »	2905.44	1	1680	1	51657.1	1361.38
23 »	1714.67	1	1600	1	551554	821.856

Figure R9.2 Predicted Values Added to the Data Set

R9.4.3 Listing Basic Partial Effects

In most of *LIMDEP*'s models, the coefficients are not the partial effects of interest. These are computed separately after estimation. Most estimators provide a basic set of results for partial effects by adding

; Partial Effects

to the model command. These will be provided in a second set of results. (In previous versions of *LIMDEP* and *NLOGIT*, the command was **; Marginal Effects**. This form is still supported, and has the same meaning in the current versions of *LIMDEP* and *NLOGIT*.)

For our tobit model from Section R9.1, the following additional output is presented after the main model results:

```
-----------------------------------------------------------------------------
Partial derivatives of expected val. with
respect to the vector of characteristics.
They are computed at the means of the Xs.
Observations used for means are All Obs.
Conditional Mean at Sample Point 613.5799
Scale Factor for Marginal Effects   .5944
--------+--------------------------------------------------------------------
        |     Partial      Standard            Prob.      95% Confidence
       Y|      Effect        Error       z    |z|>Z*          Interval
--------+--------------------------------------------------------------------
      WA|   -20.1760***     4.20411    -4.80  .0000     -28.4159   -11.9361
      WE|   -5.58460       13.13702     -.43  .6708    -31.33268   20.16348
      WW|    118.383***     9.00549    13.15  .0000      100.733    136.033
     KL6|   -497.910***    69.32012    -7.18  .0000     -633.775   -362.045
    K618|   -61.7740***    23.47397    -2.63  .0085    -107.7821   -15.7659
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
Fixed parameter ... is constrained to equal the value or
had a nonpositive st.error because of an earlier problem.
-----------------------------------------------------------------------------
```

The partial effects computed with this feature are appropriate for single index models such as the probit, logit and tobit models that do not contain nonlinear terms or interaction terms. For more intricate models, and to obtain simulations and analyses of partial effects, you will use the post estimation command **PARTIAL EFFECTS**. This command is detailed in Chapter R11.

NOTE: *LIMDEP* now provides a large set of tools for obtaining appropriate partial effects in models, for example that contain interaction terms and nonlinear parts, and for functions that you define instead of using the conditional mean.

R9.4.4 Hypothesis Tests and Restrictions

You can test hypotheses as part of the model command. The model results will contain the results of the test with the other output. Chapter R13 describes how to specify hypothesis tests and restrictions in model commands. The following shows two simple examples:

Example 1: Joint test of two restrictions. The tobit model command is modified as follows:

```
NAMELIST    ; x = one,wa,we,ww,kl6,k618 $
TOBIT       ; Lhs = y ; Rhs = x
            ; test: wa = 0, we = 0 $
```

The command specifies two restrictions to be tested jointly, the coefficient on *wa* equals zero and the coefficient on *we* equals zero. The request for a joint test is indicated by separating the specifications with a comma. The model output result is as follows (the estimation results are not changed, so they are omitted).

```
-----------------------------------------------------------------------------
Limited Dependent Variable Model - CENSORED
Dependent variable                    Y
Log likelihood function     -3817.29976
Estimation based on N =    753, K =   7
Inf.Cr.AIC  = 7648.600 AIC/N =   10.158
Threshold values for the model:
Lower=     .0000     Upper=+infinity
LM test [df] for tobit=    316.766[  6]
Normality Test, LM    =     40.989[  2]
ANOVA  based fit measure =    .225490
DECOMP based fit measure =    .236441
Wald test of  2 linear restrictions  ←
Chi-squared =      22.89, P value =  .00001
--------+--------------------------------------------------------------------
```

In addition to the standard results, the results of the Wald test of the two restrictions are shown. This is one hypothesis of two restrictions.

Example 2: Separate test of two restrictions. The tobit model command is modified as follows:

```
NAMELIST    ; x = one,wa,we,ww,kl6,k618 $
TOBIT       ; Lhs = y ; Rhs = x
            ; test: wa = 0, we = 0 | kl6 = 0, k618 = 0 $
```

The command specifies two joint restrictions. The first is the joint restriction specified earlier. The second specifies that the coefficients on the two household size variables are zero. The request for a pair of restrictions to be tested separately is made by separating the specifications with a vertical bar. The model output result is as follows: The separate hypothesis tests are now displayed after the other results.

```
-----------------------------------------------------------------------------
Limited Dependent Variable Model - CENSORED
...
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       z    |z|>Z*          Interval
--------+--------------------------------------------------------------------
...
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Chi squared tests of linear restrictions. Degrees of freedom shown
in [.]. Equals zero is implied if no specific value was given.
 1. Restriction:WA=0,WE=0
    Chi squared[ 2] =       22.891, P value =  .0000
 2. Restriction:KL6=0,K618=0
    Chi squared[ 2] =       53.877, P value =  .0000
-----------------------------------------------------------------------------
```

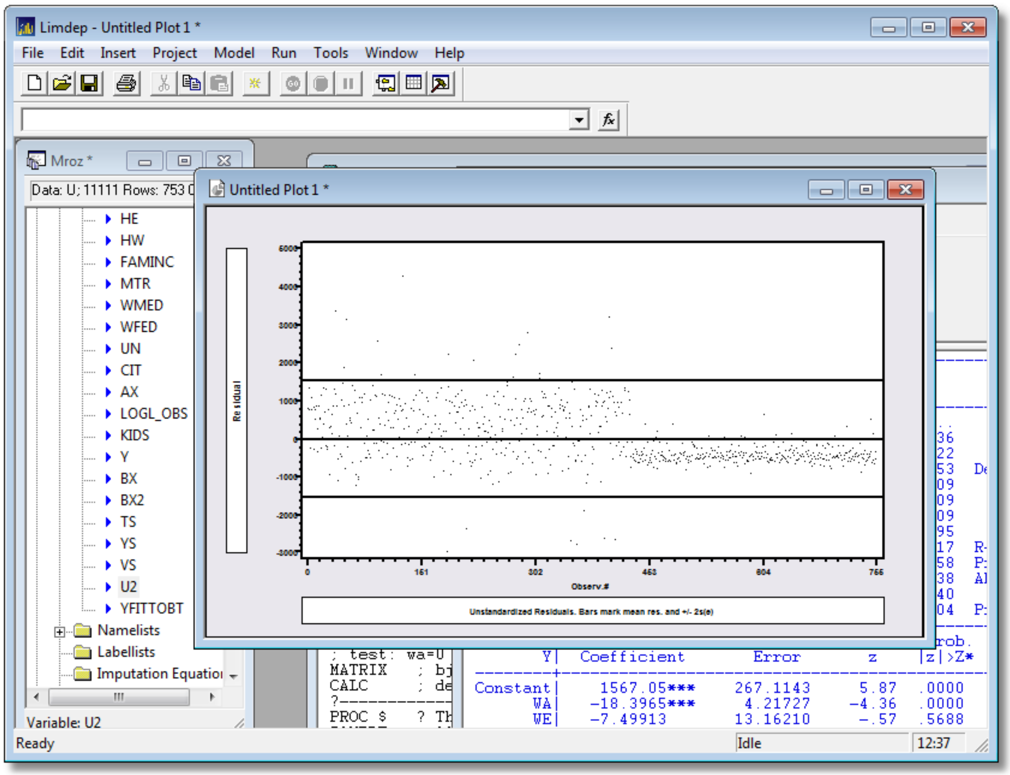
NOTE: There are several optional features and model extensions for testing and imposing restrictions. These features are described in Chapter R13.

R9.4.5 Graphical Results

Some models and commands produce graphical as well as text output. For example, to obtain a plot of residuals with a linear regression model, we would use

```
NAMELIST    ; x = one,wa,we,ww,kl6,k618 $
REGRESS     ; Lhs = y ; Rhs = x ; Plot $
```

The results shown in Figure R9.3 would result. The graphical output is displayed in a separate window. The contents of the window can be copied and moved to a document or spreadsheet. The type of output shown will vary from model to model, so they are detailed in the specific contexts in the *Econometric Modeling Guide*.



```
; test: wa=0
MATRIX     ; bj
CALC       ; de
?-----------
PROC $     ? Th
```

```
        Y|  Coefficient       Error          z   |z|>Z*
---------+----------------------------------------------
 Constant|    1567.05***   267.1143       5.87  .0000
       WA|    -18.3965***    4.21727     -4.36  .0000
       WE|    -7.49913      13.16210      -.57  .5688
```

Figure R9.3 Graphical Results Produced by a Model Command

R9.5 Suppressing Results

Model estimation commands often appear as part of iterative or repetitive calculations. Bootstrapping is a common example. In these cases, although estimation requires computation of a model, you will not be interested in seeing the results of these intermediate steps. *LIMDEP* provides two methods of suppressing output, one 'local,' that is specific to a particular model command and one 'global,' that applies to all commands submitted as a group.

R9.5.1 Suppressing Estimation Results with Quiet

You can suppress model estimation results by adding

; Quiet

to your command. This might seem counterproductive. But, you would use this if your model command were part of an iteration in which the estimation results were only to be collected and used in a later computation.

The command set below illustrates use of **; Quiet** in an estimation procedure. *LIMDEP* does not contain a built in estimator for Powell's (1986) symmetrically censored least squares (SCLS) estimator for the tobit model. But, the computations for the estimator are so simple that they can be done with a small number of basic commands. The initial **NAMELIST** and **CREATE** commands define the Rhs and Lhs variables in the model. The program can be adapted to a different application just by changing these definitions correspondingly for the data set. The rest of the commands are generic; they can be used for any data set. We have used the same specification as in the first example above. The **TOBIT** command is used to obtain starting values. It contains **; Quiet** as this is just for starting values. We will examine them more closely later. The **PROCEDURE** will be executed many times – this is the iteration. (Procedures are discussed in Chapter R19.) It computes a least squares regression that is not of separate interest in each cycle, so the **REGRESS** command in the procedure is also modified to produce no visible output with **; Quiet**.

```
?==========================================================
? Powell's (1986) symmetrically censored least squares estimator
?==========================================================
? This is the only part of the estimator that is specific to the problem. Here, the
? user defines the list of regressors and the dependent variable.
?------------------------------------------------------------------------------------------------
NAMELIST     ; x = one,wa,we,ww,kl6,k618 $
CREATE       ; y = whrs $
?------------------------------------------------------------------------------------------------
? Use the tobit MLE as starting values for beta.
?------------------------------------------------------------------------------------------------
SAMPLE       ; All $
→ TOBIT      ; Quiet; Lhs = y ; Rhs = x $
MATRIX       ; bj = b ; btobit = b ; vtobit = varb $ We compare later.
CALC         ; deltab = 1 $  Start delta large enough to begin.
?------------------------------------------------------------------------------------------------
PROCEDURE $   This procedure computes the SCLS estimator iteratively
SAMPLE       ; All $
CREATE       ; bx = x'bj ; bx2 = 2*bx ; ts = bx > 0 ; ys = Min(y,bx2) $
REJECT       ; ts = 0 $
→ REGRESS    ; Quiet ; Lhs = ys ; Rhs = x $
MATRIX       ; hj = <x'x> ;  bj1 = b ; db = bj1-bj ; bj = bj1 $
? Check for convergence using a scale free measure rather than db.
CALC         ; List(exec) ; deltab = Qfr(db,hj) $
ENDPROCEDURE $
```

```
?------------------------------------------------------------------------------------------------
EXECUTE     ; While deltab> .00001 $
?------------------------------------------------------------------------------------------------
? Estimation is finished. Get covariance matrix and show results.
?------------------------------------------------------------------------------------------------
SAMPLE      ; All $
CREATE      ; vs = (y > 0) * (y < bx2) ; u2 = ts*(ys-bx)^2 $
MATRIX      ; c = x'[vs]x ; d = x'[u2]x  ; v = <c>*d*<c> $
DISPLAY     ; Labels = x
            ; Parameters = bj
            ; Covariance = v
            ; Title = Symmetrically Censored Least Squares $
DISPLAY     ; Labels = x ; Parameters = btobit ; Covariance = vtobit
            ; Title = Maximum Likelihood Tobit Estimates $
?================================================================
```

The only results produced by this program up to the **EXECUTE** command are the trace of the convergence criterion shown below. (**EXECUTE** is discussed in Chapter R19.) Note that the iteration ends when *deltab* falls below .00001, which takes 17 iterations.

```
-->EXECUTE  ; While deltab > .00001 $
[CALC:Iteration=0001] DELTAB  =  15409.0245374
[CALC:Iteration=0002] DELTAB  =  15953.2498107
[CALC:Iteration=0003] DELTAB  =   4105.9964315
[CALC:Iteration=0004] DELTAB  =    803.3354609
[CALC:Iteration=0005] DELTAB  =    153.0440572
[CALC:Iteration=0006] DELTAB  =     33.2735713
[CALC:Iteration=0007] DELTAB  =      6.8368188
[CALC:Iteration=0008] DELTAB  =      1.4082200
[CALC:Iteration=0009] DELTAB  =       .3063135
[CALC:Iteration=0010] DELTAB  =       .0687631
[CALC:Iteration=0011] DELTAB  =       .0157467
[CALC:Iteration=0012] DELTAB  =       .0036514
[CALC:Iteration=0013] DELTAB  =       .0008487
[CALC:Iteration=0014] DELTAB  =       .0002000
[CALC:Iteration=0015] DELTAB  =       .0000475
[CALC:Iteration=0016] DELTAB  =       .0000114
[CALC:Iteration=0017] DELTAB  =       .0000027
```

R9.5.2 Suppressing All Results with SILENT

If you are using a bootstrap estimator, or searching over a parameter value as you estimate a model repeatedly, you may want to suppress the model results while you accumulate a statistic or a matrix in the background. In the previous example, the trace in **CALC** is the only visible result, and we might have been uninterested in this as well.

The command **SILENT** is used in the editor, in a procedure or in an input file for this purpose. In the SLCS example above, we used **; Quiet** in the two model commands to suppress the intermediate estimation results.

We could have used

```
SILENT
EXECUTE      ; while deltab > .00001 $
```

instead to suppress all results including the trace produced by the **CALC** command.

For example, consider the following set of commands which tests whether the set of coefficients in a regression model are the same across 10 firms using a likelihood ratio test. (This is based on the Grunfeld.lpj data set provided with the program.) For the homogeneity test, we compute the regression model for all 200 observations, then for each of 10 firms. The test statistic is two times the sum of the log likelihoods for the subsamples minus two times the log likelihood for the pooled data.

```
SILENT
SAMPLE       ; 1-200 $
REGRESS      ; Lhs = i ; Rhs = one,f,c $
CALC         ; sumlogl = -2 * logl ; company = 0 $
PROCEDURE $
CALC         ; i1 = 20 * (company-1) + 1 ; i2 = i1 + 19 $
SAMPLE       ; i1 - i2 $
REGRESS      ; Lhs = i ; Rhs = one,f,c $
CALC         ; sumlogl = sumlogl + 2 * logl $
ENDPROCEDURE $
EXECUTE      ; company = 1,10 $
CALC         ; List ; chisq = sumlogl $
```

This procedure estimates 11 regression models. Our only interest is in the statistic *sumlogl* that is accumulated. So, before executing the block of commands, we use **SILENT** to suppress all of the output from the commands. When we are finished, we use **CALC** to retrieve the statistic.

Note that silent execution is only for the duration of the current block of commands being executed. A block of commands is executed by highlighting it in the editing window then clicking GO. Once the block is finished, the switch is automatically turned off. This prevents you from leaving the switch on. Do note what this implies for execution. Suppose that these two lines are on the screen in your editor:

```
SILENT
REGRESS      ; Lhs = … etc. $
```

If you highlight only the **SILENT** command, click GO, then highlight the **REGRESS** command and click GO a second time, the **REGRESS** command will *not* be executed silently. If you highlight *both* lines then click GO once, the **REGRESS** command *will* be executed silently. We will examine **EXECUTE** in a later chapter. For now, note that you can localize the **SILENT** command by using

```
EXECUTE      ; Silent ; … the rest of the setup $
```

R9.6 The Review Window – Tables of Model Results

In normal usage, model results are displayed one at a time. They may be recovered later from the output file (or window) in a word processor if you wish to collect them in tables. You can also assemble tables of results as you do your estimation, and send tables that combine results to the output file. You can retain a 'stack' of up to 10 model results at a time by adding

```
                    ; Table = up to eight character label
```

to any model command. This adds the model results to a stack of the last 10 tabled models. Additional models push the stack downward. Thus, if you table an 11th model, it pushes model 1 off the stack, and this one becomes model 10. To review the results in the stack, select Tools:Review Tables or double click any of the models listed in the Output:Tables group in the project window. You can then select any of the model results listed to open the Review Tables dialog box. This feature allows you to review the model results.

For an example, consider the following sequence of commands:

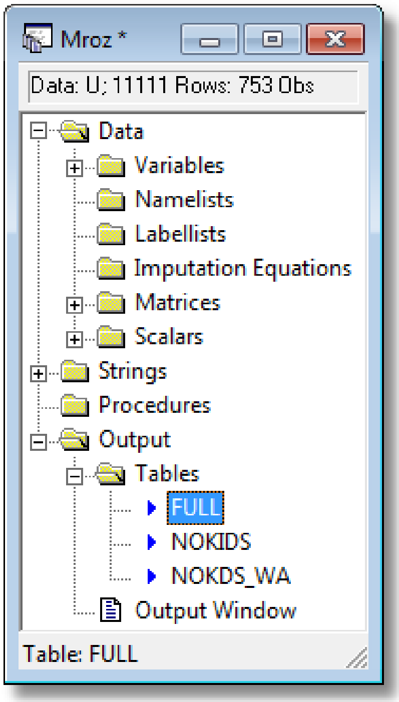
```
TOBIT       ; Lhs = whrs ; Rhs = one,wa,we,ww,kl6,k618
            ; Table = Full $
TOBIT       ; Lhs = whrs ; Rhs = one,wa,we,ww
            ; Table = Nokids $
TOBIT       ; Lhs = whrs ; Rhs = one,we,ww
            ; Table = Nokds_wa $
```

The project window is updated after estimation as shown in Figure R9.4. Double clicking any of the names in the Tables folder invokes the model review dialog box shown in Figure R9.5. You can produce brief summary tables with your command processor. The command is

```
REVIEW      ; Title = … the title you'd like to give the table
            ; Model = list of up to three model names $
```

You may have one, two, or three model specifications in the command. An example is shown below.

```
REVIEW      ; Title = Three Tobit Specifications
            ; Model = Full,NoKids,NoKds_We $
```



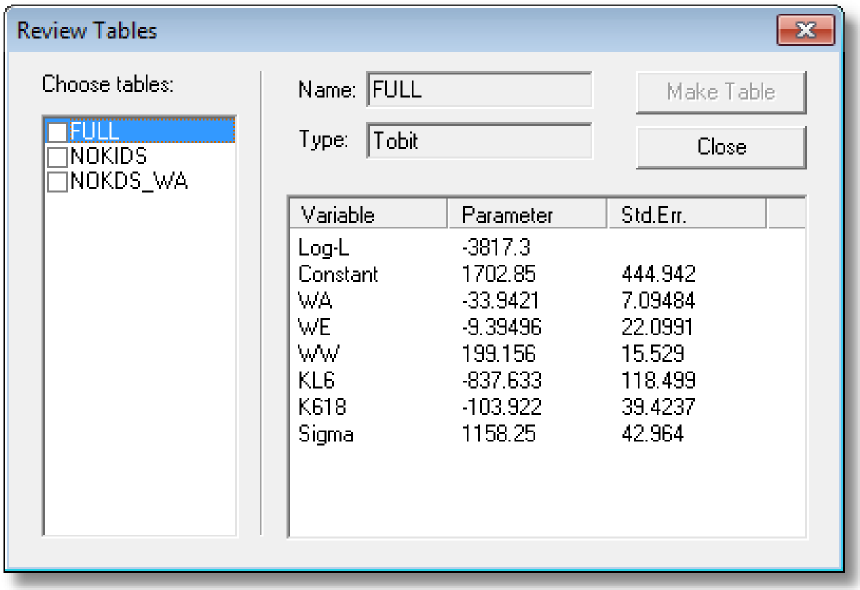
Mroz *

Data: U; 11111 Rows: 753 Obs

- Data
  - Variables
  - Namelists
  - Labellists
  - Imputation Equations
  - Matrices
  - Scalars
- Strings
- Procedures
- Output
  - Tables
    - FULL
    - NOKIDS
    - NOKDS_WA
  - Output Window

Table: FULL

Figure R9.4 Project Window with Model Table



Review Tables

Choose tables:

- FULL
- NOKIDS
- NOKDS_WA

Name: FULL

Type: Tobit

Variable	Parameter	Std.Err.
Log-L	-3817.3	
Constant	1702.85	444.942
WA	-33.9421	7.09484
WE	-9.39496	22.0991
WW	199.156	15.529
KL6	-837.633	118.499
K618	-103.922	39.4237
Sigma	1158.25	42.964

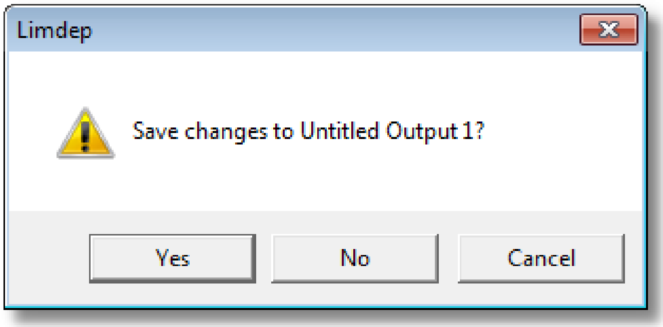
Figure R9.5 Review Tables Dialog Box

```
+-----------------------------------------------------------------------------+
|                        Three Tobit Specifications                            |
+----------+----------------------+----------------------+----------------------+
|          |         FULL         |        NOKIDS        |       NOKDS_WA       |
+----------+----------------------+----------------------+----------------------+
| Variable |  Parameter|  t-ratio|  Parameter|  t-ratio|  Parameter|  t-ratio|
+----------+-----------+---------+-----------+---------+-----------+---------|
| Constant |  1702.8550|    3.827|   370.1806|     .932|     2.4064|     .009|
| WA       |   -33.9421|   -4.784|    -7.8260|   -1.272|           |         |
| WE       |    -9.3950|    -.425|   -22.7934|   -1.004|   -19.8272|    -.878|
| WW       |   199.1557|   12.825|   221.5582|   13.725|   221.3327|   13.712|
| KL6      |  -837.6332|   -7.069|           |         |           |         |
| K618     |  -103.9224|   -2.636|           |         |           |         |
| Sigma    |  1158.2500|   26.959|  1212.3680|   26.817|  1212.8740|   26.816|
| Log-L    | -3817.3000|         | -3846.1880|         | -3847.0000|         |
+----------+-----------+---------+-----------+---------+-----------+---------+
```

R9.7 Output Files

All of your model results are being accumulated in the output window, which will be prominent on your desktop. When you exit *LIMDEP* to end your session, you will be asked if you wish to save the contents of the output window in a file – the query will typically appear as shown in Figure R9.6. You can at this point create an output file for the session just by clicking Yes. You will have an opportunity to name the file with any filename you choose.

WARNING: Output files and command files are both saved with the .lim extension. You will need to make careful note of which files you save are which type.



Save changes to Untitled Output 1?

Figure R9.6 Query to Save Output Window

You may also open a separate output file for a session or a part of a session at any time with the command

OPEN ; Output = filename $

Once an output file is opened, all output that appears in the output window is duplicated in the file. You may close the current output file at any time either by opening a new one or with the command

CLOSE

You may add a title that will appear at the top of each ‘page’ in the output file with

```
TITLE          ; any string of up to 72 characters $
```

This title will appear at the top of each page in the output file until you give a new title command. (Pages in the output file are only relative. At certain points, the output is delimited with a banner that displays useful information about changes in the sample, beginning of a model estimation procedure, etc.)

A title string may insert the value contained in a scalar with the syntax

```
TITLE          = … \sname...
```

The ‘\sname’ signifies that the current value contained in the scalar with that name is to be inserted into the title at that point. For an example, you might be plotting a function of a few values;

```
CALC           ; theta = 0.545 $
FPLOT          ; ... ; Title = Function Plot for Theta = \theta $
CALC           ; theta = 0.875 $
FPLOT          ; ... ; Title = Function Plot for Theta = \theta $
```

A similar device may be used to insert a variable name in an indexed namelist with

```
TITLE          = …\namelist:index
```

You might use this in a loop with indexing over variables in a list. For example, the following computes the same regression for several variables, and plots the residuals:

```
NAMELIST       ; y = y1,y2,y3 $
PROCEDURE $
REGRESS        ; Lhs = y: i ; Rhs = x ; Res = e $
PLOT           ; Rhs = e ; Title = Residuals for Regression of \y:i on x $
ENDPROCEDURE $
EXECUTE        ; i = 1,3 $
```

R9.7.1 Transporting Output Results to Word Processors

You can lift blocks text from *LIMDEP*’s output window and drop it into any word processing program (or the reverse). You can then edit the output in your word processor.

TIP: When you copy from the output window into *Word*, *LIMDEP*’s font formatting is typically lost. We find the best results by changing the font to Courier New, size 9 after it is ‘pasted.’

You cannot edit the text in the output window – you can highlight and delete it, but that is the only direct editing function in the output window. But, you can copy text from the output window to any text editing window. You can also have multiple text editing windows open. It might be convenient to have one text window open for commands and another for editing output.

R9.7.2 Exporting Statistical Results from *LIMDEP*

You can export your statistical results to other packages. The preceding shows how to produce output files in text format that can be copied directly into word processing programs. With copy/paste, you can extract matrix results and drop them directly into spreadsheet programs. You can also export your results more formally to any program that can accept the 'comma separated values,' or CSV format, such as *Excel*. The file that *LIMDEP* creates can be read directly, without any further manipulation on your part. Setting it up requires a few steps, as shown below.

Step 1. Open the file that will contain the results to be exported.

This will be a .csv (comma separated values) file. Use the following *LIMDEP* **OPEN** command:

```
OPEN      ; Export = …<filename>.csv $
```

You must open the file with extension .csv for this operation to succeed. *LIMDEP* does not check this file setup for you – the program assumes that the file is opened correctly.

Step 2. Use the ; Export specification in your model commands.

In specific model commands that you wish to export, use the model option **; Export** to put a table of coefficients, etc. in the export file. You may also use **; Title = up to 80 characters** to put a line of text at the top of the results. For some other specific commands, you can use

MATRIX	**; Export = list of matrices $**	puts a list of matrices in the file.
DSTAT	**; Export ; Rhs = ... $**	copies the results to the CSV file.
CALC	**; Export = list of scalars $**	copies scalars to the file.

Step 3. Close the file before you try to use it.

When you are finished exporting results to the file, use

```
CLOSE     ; Export $
```

to end accumulation of results in the file.

After this file is created, you can now export your results to *Excel* just by double clicking the file name in any context, such as Windows Explorer. There are two possible conflicts to be wary of:

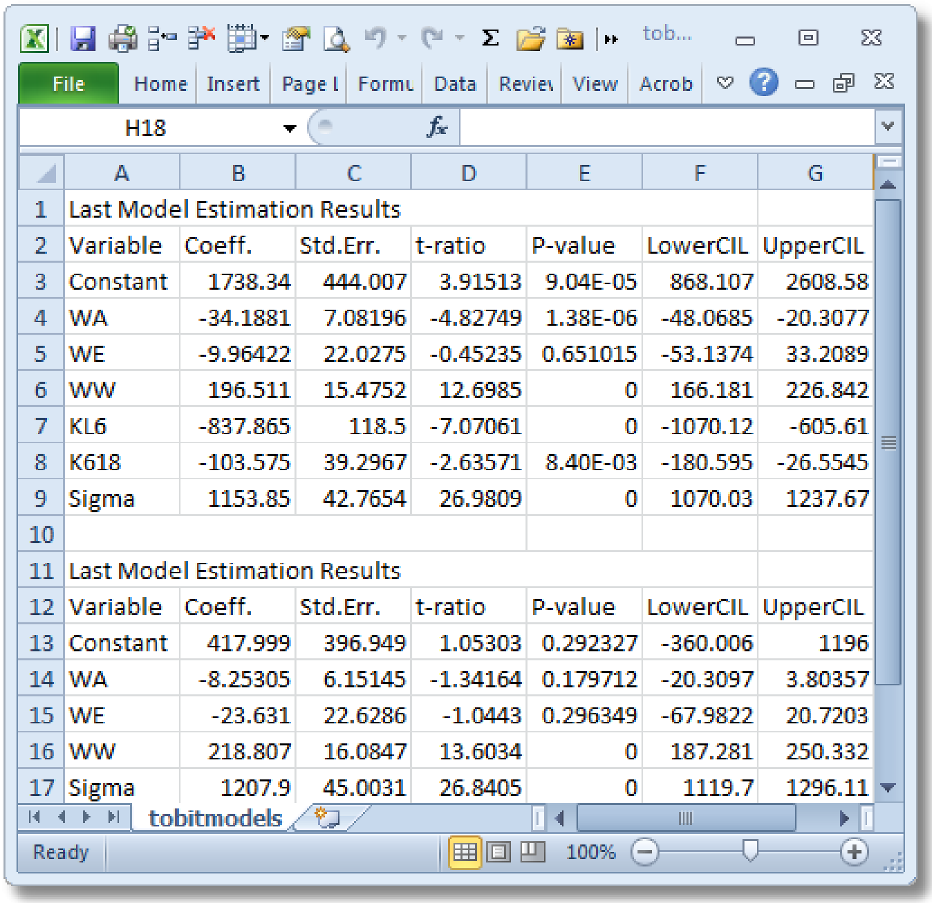
- The file cannot be reopened. If you repeat an **OPEN ; Export = name $** command, the original file is erased and a new one with that name is created.

- Do not use this file, e.g., by *Excel*, until you exit *LIMDEP*, even if you have used a **CLOSE** command to close the file.

An example follows: We create the file in *LIMDEP*.

```
OPEN        ; Export = "C:\...\tobitmodels.csv" $
TOBIT       ; Lhs = whrs ; Rhs = one,wa,we,ww,kl6,k618 ; Export $
TOBIT       ; Lhs = whrs ; Rhs = one,wa,we,ww          ; Export $
CLOSE       ; Export $
```

We then open the file in *Excel*:



	A	B	C	D	E	F	G
1	Last Model Estimation Results						
2	Variable	Coeff.	Std.Err.	t-ratio	P-value	LowerCIL	UpperCIL
3	Constant	1738.34	444.007	3.91513	9.04E-05	868.107	2608.58
4	WA	-34.1881	7.08196	-4.82749	1.38E-06	-48.0685	-20.3077
5	WE	-9.96422	22.0275	-0.45235	0.651015	-53.1374	33.2089
6	WW	196.511	15.4752	12.6985	0	166.181	226.842
7	KL6	-837.865	118.5	-7.07061	0	-1070.12	-605.61
8	K618	-103.575	39.2967	-2.63571	8.40E-03	-180.595	-26.5545
9	Sigma	1153.85	42.7654	26.9809	0	1070.03	1237.67
10							
11	Last Model Estimation Results						
12	Variable	Coeff.	Std.Err.	t-ratio	P-value	LowerCIL	UpperCIL
13	Constant	417.999	396.949	1.05303	0.292327	-360.006	1196
14	WA	-8.25305	6.15145	-1.34164	0.179712	-20.3097	3.80357
15	WE	-23.631	22.6286	-1.0443	0.296349	-67.9822	20.7203
16	WW	218.807	16.0847	13.6034	0	187.281	250.332
17	Sigma	1207.9	45.0031	26.8405	0	1119.7	1296.11

Figure R9.7 *LIMDEP* Results Exported to *Excel*

If you are computing many different models or exporting results during an iteration, you might want to have several export files, say one for each iteration. You can index the file name for Export as follows:

```
OPEN        ; Export = <name…># number #<name…> $
```

where the number is the index you want in the name.

For example, the following would create ten export files:

```
PROCEDURE $
OPEN        ; Export = "C:\models\ExportResults#i#" $
            … model commands
ENDPROCEDURE $
EXECUTE     ; i = 1,10 $
```

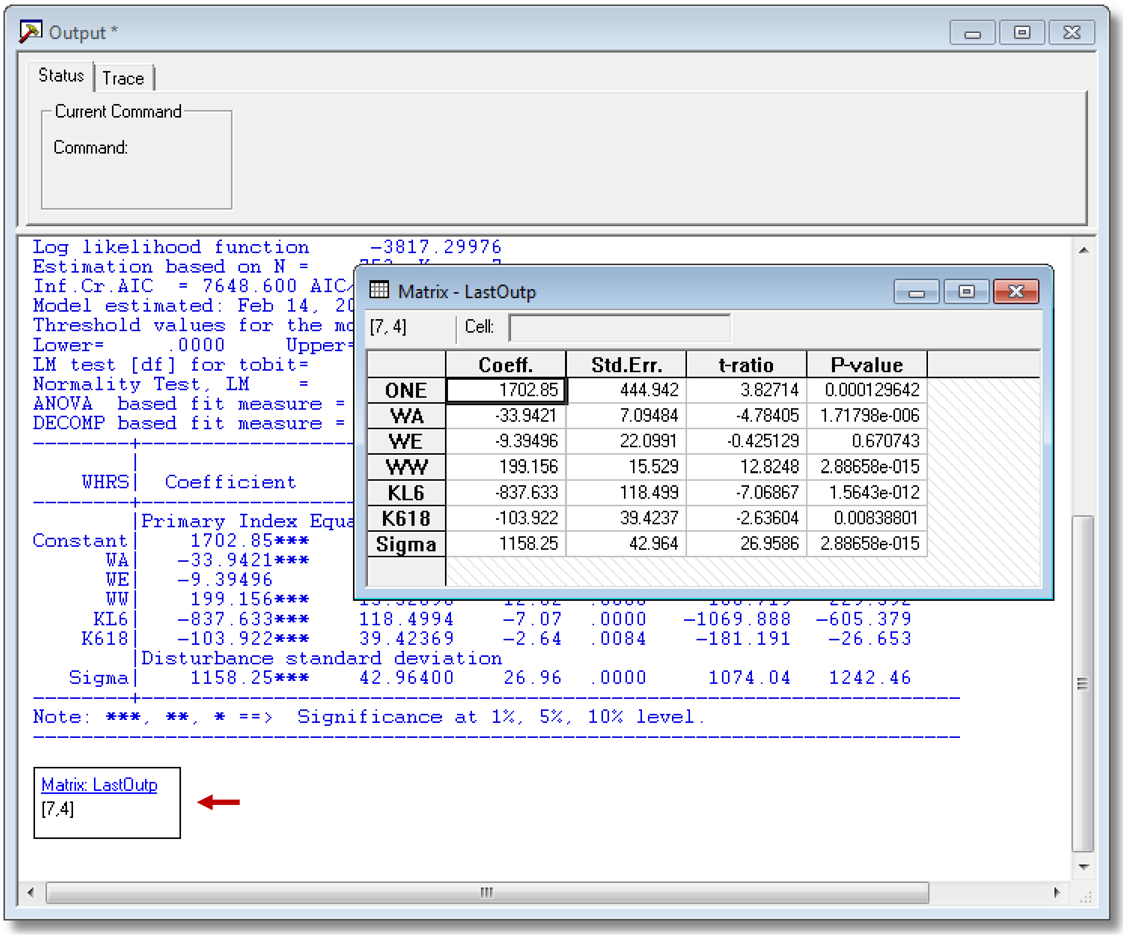
The files would be named ExportResults1, ExportResults2, etc. Note, although the program creates several files, only one of them at a time is open.

R9.7.3 The Last Model Output

The results in Figure R9.8 are produced by the command

```
TOBIT          ; Lhs = whrs ; Rhs = one,wa,we,ww,kl6,k618
               ; Matrix $
```

The **; Matrix** specification requests the embedded matrix object c which is shown at the lower left of the window in Figure R9.8. Double clicking the object opens the Matrix:LastOutp window containing the output. If the **; Matrix** switch is omitted from the command, this extra matrix output does not appear in the output.



```
Log likelihood function      -3817.29976
Estimation based on N =
Inf.Cr.AIC  = 7648.600 AIC/
Model estimated: Feb 14, 20
Threshold values for the mo
Lower=      .0000     Upper=
LM test [df] for tobit=
Normality Test, LM     =
ANOVA  based fit measure =
DECOMP based fit measure =
--------+-------------------
        |
    WHRS|  Coefficient
--------+-------------------
        |Primary Index Equa
Constant|    1702.85***
      WA|    -33.9421***
      WE|    -9.39496
      WW|     199.156***
     KL6|   -837.633***     118.4994     -7.07   .0000    -1069.888   -605.379
    K618|   -103.922***     39.42369     -2.64   .0084    -181.191    -26.653
        |Disturbance standard deviation
   Sigma|    1158.25***     42.96400     26.96   .0000     1074.04    1242.46
--------+-----------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
--------------------------------------------------------------------------
```

Matrix: LastOutp
[7,4]

	Coeff.	Std.Err.	t-ratio	P-value
ONE	1702.85	444.942	3.82714	0.000129642
WA	-33.9421	7.09484	-4.78405	1.71798e-006
WE	-9.39496	22.0991	-0.425129	0.670743
WW	199.156	15.529	12.8248	2.88658e-015
KL6	-837.633	118.499	-7.06867	1.5643e-012
K618	-103.922	39.4237	-2.63604	0.00838801
Sigma	1158.25	42.964	26.9586	2.88658e-015

Figure R9.8 Last Model Output as an Embedded Matrix

By clicking the upper left (blank) cell in this or any other matrix that *LIMDEP* displays, you will highlight the entire matrix. You can then use edit copy/paste to export this output to another program, such as *Excel*. The material that is moved to *Excel* is the same as that produced by **; Export** in the previous section.

R10: Robust Covariance Matrices and Clustering

R10.1 Robust Covariance Matrix for Pooled Models

Robust covariance matrices are used to estimate asymptotic covariance matrices for estimators when model assumptions may not be met. Familiar examples include the White estimator (see Section E7.9.1) for heteroscedasticity in regression and the Newey-West estimator (see Section E7.9.2) for autocorrelation. For cross sections and 'pooled' maximum likelihood estimators, the counterpart to White is the 'sandwich' estimator,

$$\mathbf{V} = \mathbf{H}^{-1}\mathbf{OPG}\mathbf{H}^{-1}$$

Where **H** is the negative of the second derivatives and **OPG** is the sum of the outer products of the gradients of the terms in the log likelihood function,

$$\mathbf{OPG} = \sum_{i=1}^{n} \left[\frac{\partial \ln f_i(\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}}\right]\left[\frac{\partial \ln f_i(\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}}\right]'.$$

In some settings, V can overcome a misspecification of the model, for example, in the presence of unmeasured heterogeneity in the conditional mean function of the Poisson regression model. In other cases, researchers routinely use this estimator under the assumption that it compensates for other unspecified types of misspecification.

The robust covariance matrix is provided explicitly for a few models in *LIMDEP*, such as **LOGIT** and **POISSON**, by placing

```
; Robust
```

in the command. For those for which is not explicitly provided, there is a way to 'trick' *LIMDEP* into computing it anyway. The 'clustering' estimator discussed in the next section is provided for all estimators in *LIMDEP* (that are based on the likelihood function). The cluster estimator when every cluster has one observation is identical to this sandwich estimator. So, you can use

```
; Cluster = 1
```

with any MLE to obtain this robust estimator.

To illustrate, we obtain the standard estimator and compare it to the robust estimator for a probit model. The application is based on Mroz.lpj. The basic model command is

```
PROBIT      ; Lhs = lfp ; Rhs = one,wa,we,kl6,k618 $
```

The Lhs variable is the binary variable for labor force participation. The Rhs variables are age, education, kids under six and kids six to eighteen. The standard results are shown below. (Some results are omitted.)

```
-----------------------------------------------------------------------------
Binomial Probit Model
...
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
     LFP|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|     .62379         .46637     1.34  .1810     -.29028   1.53786
      WA|    -.03827***      .00746    -5.13  .0000     -.05288   -.02366
      WE|     .12003***      .02219     5.41  .0000      .07655    .16351
     KL6|    -.88612***      .11242    -7.88  .0000    -1.10646   -.66577
    K618|    -.05569         .04009    -1.39  .1647     -.13426    .02287
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

When **; Robust** is added to the command, the parameter estimates are the same, since the correction only adjusts the standard errors.

```
-----------------------------------------------------------------------------
Binomial Probit Model
...
Robust VC=<H>G<H> used for estimates.
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
     LFP|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|     .62379         .45877     1.36  .1739     -.27538   1.52297
      WA|    -.03827***      .00742    -5.16  .0000     -.05281   -.02373
      WE|     .12003***      .02216     5.42  .0000      .07659    .16347
     KL6|    -.88612***      .11650    -7.61  .0000    -1.11445   -.65779
    K618|    -.05569         .04106    -1.36  .1750     -.13617    .02479
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

If **; Cluster = 1** is added to the command, instead, the results are the same, but the indicator of the 'Robust VC' is replaced by a line of text before the results,

```
+-----------------------------------------------------------------------+
| Robust covariance matrix, <H>*OPG*<H> is used for the estimator.    |
+-----------------------------------------------------------------------+
```

R10.2 Using Clustering for Robust Covariance Matrices

A robust estimator based on sample clustering is available for nearly all models estimated by *LIMDEP*. The estimator that *LIMDEP* computes for the asymptotic covariance matrix of the MLE is

$$\text{Est.Asy.Var}\left[\hat{\boldsymbol{\theta}}\right] = \mathbf{V}\times\frac{C}{C-1}\sum\nolimits_{c=1}^{C}\mathbf{g}_c\mathbf{g}_c{}'\times\mathbf{V}$$

where V is the usual asymptotic covariance matrix estimator ignoring the clustering, C is the number of clusters, and

$$\mathbf{g}_c = \sum\nolimits_{i=observations\ in\ the\ cluster}\mathbf{g}_{ic}$$

This is the outer product estimator in which observations are the sums of observations in the cluster. See below for technical details on this estimator. In order to use this estimator, it is necessary only to identify the cluster in the model command. Use one of the following

; Cluster = n$_c$ where n_c is the fixed number of observations in each cluster.

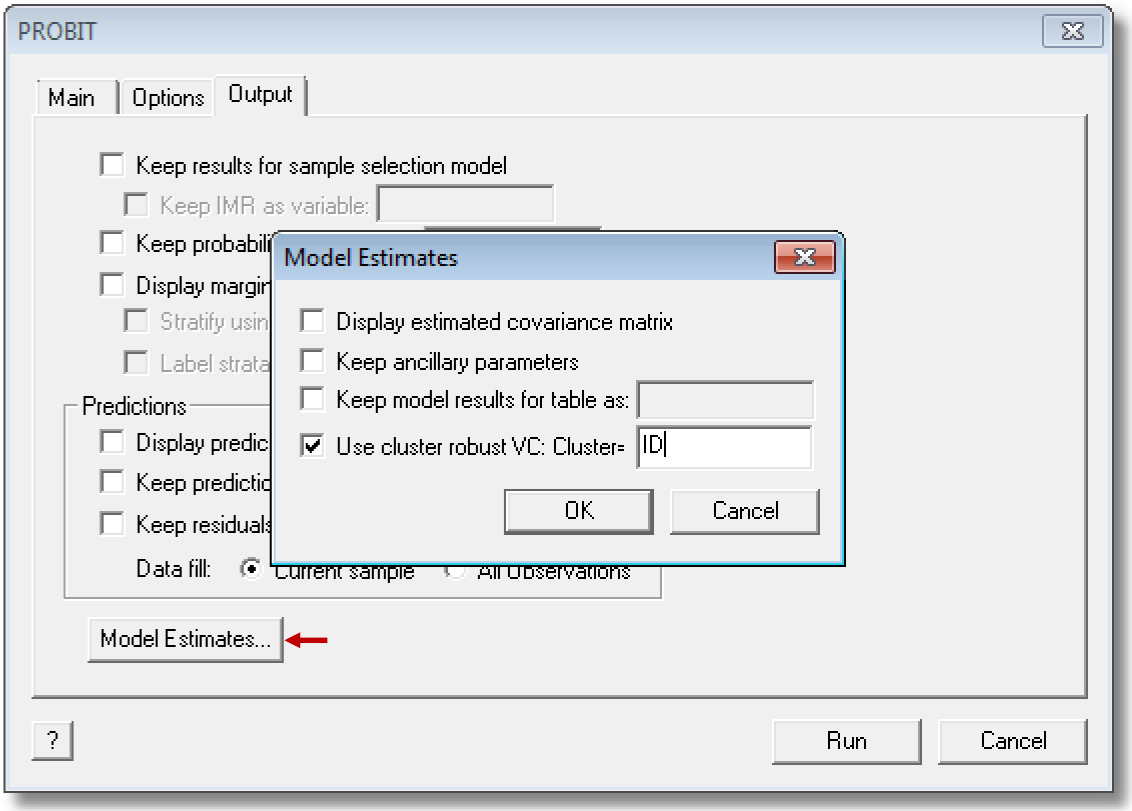
Use this form if every cluster has the same number of observations. Alternatively, if the number of observations in the clusters is different, you must provide some sort of identification variable (not a count variable), such as might be used in the panel data estimator for the linear model. This form is

; Cluster = name of ID variable

This arrangement resembles a panel data setup. The variable may be any distinct (numeric) indicator of the group; it need not be a consecutive set of integers. A third possibility is that you have a variable which gives the number of observations per group, as in most of *LIMDEP*'s panel estimators, but you do not have the group ID number that you need. You can create the ID variable with

```
CREATE      ; groupID = GroupNmbr (count variable) $
```

The option for clustering is offered in the command builders for all the nonlinear model routines in the Model Estimates submenu. This will differ a bit from model to model. The one for the probit model is shown below in Figure R10.1. The Model Estimates dialog box is selected at the bottom of the Output page, then the clustering is specified in the next dialog box.


PROBIT

Main | Options | Output

Keep results for sample selection model

Keep IMR as variable:

Keep probabili

Display margin

Stratify usin

Label strata

Predictions

Display predic

Keep predictio

Keep residuals

Data fill: [✓ Current sample] All Observations

Model Estimates...

? Run Cancel

Model Estimates

Display estimated covariance matrix

Keep ancillary parameters

Keep model results for table as:

[✓ Use cluster robust VC: Cluster=] ID

OK Cancel

Figure R10.1 Command Builder for a Probit Model

R10.2.1 Models for Which the Clustering Estimator is Supported

This procedure may be used with any model in *LIMDEP*, but the estimator is not supported for any of the panel data specifications. For some applications in *LIMDEP*, the estimator **V** is based on the first derivatives of the log likelihood and not the second. This aspect is discussed further in the technical notes to follow and noted in the documentation for specific models. In cases where the BHHH estimator is used to compute **V**, the assumptions that underlie the cluster estimator may not be met. In particular, there must be a presumption that the BHHH estimator and the negative inverse Hessian estimator converge to the same matrix. For the kinds of applications for which this cluster estimator appear to be designed, that seems very likely to be the case. It would not be the case where users sought to protect themselves against model misspecification, but that is a different issue from clustering. On the other hand, in most such cases, the parameter estimator will be inconsistent, so robust covariance matrix estimation is a moot point.

R10.2.2 An Example of the Clustering Estimator

The following compute the corrected covariance matrix for a probit estimator. The sample is the health care data healthcare.lpj. There are 27,326 observations in the data, and 7,293 groups ranging in size from one to seven. The commands are

```
PROBIT     ; Lhs = doctor ; Rhs = one,age,educ,married,hsat $
PROBIT     ; Lhs = doctor ; Rhs = one,age,educ,married,hsat ; Cluster = id $
```

```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               DOCTOR
Log likelihood function    -16639.15794
Restricted log likelihood  -18019.55173
Chi squared [   4 d.f.]      2760.78759
Significance level               .00000
McFadden Pseudo R-squared      .0766053
Estimation based on N =  27326, K =   5
Inf.Cr.AIC  =33288.316 AIC/N =    1.218
Hosmer-Lemeshow chi-squared =  20.38745
P-value=  .00897 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|    1.35413***      .06315    21.44  .0000     1.23036   1.47791
     AGE|     .00849***      .00075    11.30  .0000      .00702    .00996
    EDUC|    -.01544***      .00346    -4.46  .0000     -.02223   -.00866
 MARRIED|     .00818         .01905      .43  .6678     -.02917    .04552
HSAT|    -.17506***      .00396   -44.25  .0000     -.18281   -.16730
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
+-----------------------------------------------------------------------+
| Covariance matrix for the model is adjusted for data clustering.      |
| Sample of  27326 observations contained   7293 clusters defined by    |
| variable ID       which identifies by a value a cluster ID.           |
+-----------------------------------------------------------------------+
-----------------------------------------------------------------------------
Binomial Probit Model
... (this part is identical)
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|    1.35413***      .08506    15.92  .0000     1.18742   1.52085
     AGE|     .00849***      .00100     8.49  .0000      .00653    .01045
    EDUC|    -.01544***      .00485    -3.18  .0015     -.02495   -.00593
 MARRIED|     .00818         .02523      .32  .7459     -.04128    .05763
HSAT|    -.17506***      .00490   -35.71  .0000     -.18467   -.16545
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

R10.2.3 Technical Details on the Clustering Estimator

The literature contains a variety of robust covariance matrices for maximum likelihood estimators. Most of these take the form of the ‘sandwich estimator,’

$$\text{Est.Var}\left[\hat{\boldsymbol{\theta}}\right] = (\textbf{-H})^{-1}\times(\mathbf{G'G})\times(\textbf{-H})^{-1}$$

where **H** is an estimator of the second derivatives matrix of the log likelihood function and **G** is an $n\times K$ matrix whose ith row is the vector of partial derivatives of the log density for the ith observation with respect to the K parameters. (Thus, the gradient of the log likelihood for independent observations is **G′i** where **i** is a column of ones.) The White estimator for least squares in the presence of unspecified heteroscedasticity is a well-known application. The following describes an application to ‘clustered’ data in which a sample of n observations is composed of C ‘clusters,’ each of which contains n_c observations, $c = 1,...,C$ – the number may differ across clusters. This is somewhat related to panel data treatments, though users should not take the analogy very far, as none of the treatment described here takes formal account of a panel structure in a data set. Our technical presentation is fairly brief. (The settings in which data might be ‘clustered’ but are not appropriate for a formal panel data treatment are discussed there.)

The robust ‘sandwich’ estimator used in many applications arises in the following manner: The asymptotic distribution of the maximum likelihood estimator derives from the following fundamental results for a regular estimator:

$$\sqrt{n}\left[\hat{\boldsymbol{\theta}}-\boldsymbol{\theta}\right]=\left[-\frac{1}{n}\hat{\mathbf{H}}^{-1}\right]\sqrt{n}\left(\frac{1}{n}\sum\nolimits_{i=1}^{n}\hat{\mathbf{g}}_i\right) + \text{higher order terms which vanish as } n\rightarrow\infty,$$

where **H** is the negative second derivatives matrix, which will be a sum, $\mathbf{g}_i$ is the ith term in the first derivative vector and the carats indicate computation at the MLE. Under the assumptions of the model, the matrix in square brackets, which is the mean of a sample, converges to its population counterpart, a finite positive definite matrix, while the mean in rounded brackets converges in probability to zero. The limiting distribution of the statistic on the left hand side of the equation is normal (see Greene (2012) for discussion) with mean zero and variance equal to the variance of the product on the right hand side. The asymptotic variance of the MLE, $\hat{\boldsymbol{\theta}}$ will then be $1/n$ times the resulting limiting variance (assuming it exists, which we are doing here). As noted, the matrix in square brackets converges to something, a matrix we’ll call $\mathbf{B}^{-1}$. The mean in round brackets is assumed to be (at least as the sample increases in size) the mean of a random sample with a finite variance. In fact, for regular ML problems, that matrix is **B**, but rather than assume that the problem is properly specified, we will leave this true variance unspecified, and use a consistent estimator of it. The sum in the round brackets is the derivative of the log likelihood which we have equated to zero to obtain the MLE. Dividing it by n, we obtain $\bar{\mathbf{g}}$, the mean of a sample. Since the true mean is known to be zero, we can estimate the variance of the mean by $1/n$ times the sample variance, which would be $1/n$ times $1/n$ times the sum of squares. (Whether this should be divided by n-1 rather than n is debatable. The result which would dictate this only holds as $n\rightarrow\infty$.) Since this is a vector of variables, rather than just one, we use the sum of outer products. Thus, combining our results, we obtain the estimator

$$\text{Est.Asy.Var.}\left(\frac{1}{n}\sum\nolimits_{i=1}^{n}\hat{\mathbf{g}}_i\right) = \frac{1}{n}\frac{1}{n}\sum\nolimits_{i=1}^{n}\hat{\mathbf{g}}_i\hat{\mathbf{g}}_i'.$$

Combining terms, we obtain our asymptotic variance estimator,

$$\text{Est.Asy.Var}\sqrt{n}\left[\hat{\boldsymbol{\theta}}-\boldsymbol{\theta}\right]=\left[-\frac{1}{n}\hat{\mathbf{H}}^{-1}\right]\left(\sqrt{n}\right)^{2}\frac{1}{n}\frac{1}{n}\left(\sum_{i=1}^{n}\hat{\mathbf{g}}_i\hat{\mathbf{g}}_i'\right)\left[-\frac{1}{n}\hat{\mathbf{H}}^{-1}\right]$$

If the model is properly specified, the center term converges to the inverse of each of the outer terms, which leaves the usual result for the asymptotic variance of the MLE, namely the inverse of the negative of the Hessian. Our estimator of the asymptotic variance of the MLE, itself, is obtained by dividing the resulting expression by *n*. After several cancellations, this produces the familiar sandwich estimator for maximum likelihood estimators,

$$\text{Est.Asy.Var.}\left[\hat{\boldsymbol{\theta}}\right]=\left[-\hat{\mathbf{H}}\right]^{-1}\left(\sum_{i=1}^{n}\hat{\mathbf{g}}_i\hat{\mathbf{g}}_i'\right)\left[-\hat{\mathbf{H}}\right]^{-1}.$$

It is a valid estimator under the following assumptions:

1. The original expansion is valid; that is the first derivatives really do converge to zero.
2. The mean of the sample estimated second derivatives – the matrix in square brackets – does converge to a finite matrix.

Finally, the clustering estimator discussed in this section is based on the idea that there is a grouping of the observations in the data set into larger observations which are connected in some fashion (correlated seems inappropriate). In this instance, the estimator is modified to produce

$$\text{Clustered Est.Asy.Var.}\left[\hat{\boldsymbol{\theta}}\right]=\left[-\hat{\mathbf{H}}\right]^{-1}\left(\frac{C}{C-1}\sum_{c=1}^{C}\mathbf{g}_c\mathbf{g}_c{}'\right)\left[-\hat{\mathbf{H}}\right]^{-1}.$$

Note that if there are very few clusters, this can produce very large standard errors. Note also the important result that this estimator does not require the MLE to converge to the parameters of interest. It only requires the MLE to converge to something. Consider for an example, the probit model with heteroscedasticity:

$$y_i^* = \boldsymbol{\beta}'\mathbf{x}_i + \varepsilon_i,\ \varepsilon_i \sim N[0, \exp(\boldsymbol{\gamma}'\mathbf{z}_i)] \quad \text{(latent structure)}$$

$$y_i = 1 \text{ if } y_i^* > 0, 0 \text{ otherwise.}$$

Suppose one 'estimates' $\boldsymbol{\beta}$ by standard probit analysis, ignoring the heteroscedasticity. Then, if $\boldsymbol{\gamma}$ is not zero, this estimator is not consistent for $\boldsymbol{\beta}$. Depending on the remaining structure of the model, and the nature of the data, it may not be consistent for anything. But, in most circumstances, this 'MLE' will converge to something; let's call it $\boldsymbol{\delta}$. Though it is less than obvious, under this assumption, the conditions of the estimator above are met, but the simple Hessian will not give the appropriate asymptotic covariance matrix. The sandwich estimator will. It must be remembered, however, that this estimator is an appropriate estimator for the asymptotic covariance matrix of an inconsistent parameter estimator. There are cases in which the probability limit of the MLE, $\boldsymbol{\delta}$ will equal the $\boldsymbol{\beta}$ of interest, such as in the Poisson model with latent heterogeneity, but there will not be very many such cases.

R10.3 Stratified and Grouped Data

This extension adds features to the **; Cluster** feature described in Section R10.2. The base case invoked by **; Cluster** changes the computation of the asymptotic covariance matrix for an estimator. The main application is maximum likelihood estimators, for which the conventional estimator of the asymptotic covariance matrix of the estimator is

$$\mathbf{V} = [\Sigma_i \mathbf{H}_i]^{-1}$$

where H_i is the sample estimate of the second derivatives matrix for the contribution of observation *i* to the log likelihood function. The so-called 'cluster estimator' uses

$$\mathbf{V}\times\mathbf{G}\times\mathbf{V} = \mathbf{V}\times[\ (C/(C\text{-}1)\Sigma_c\ (\Sigma_{i=1,Nc}\mathbf{g}_{ic})\ (\Sigma_{i=1,Nc}\mathbf{g}_{ic}')]\ \times\mathbf{V}$$

where C is the number of groups (clusters), N_c is the number of observations in group c and $\mathbf{g}_{ic}$ is the first derivative of the contribution of individual i in group c to the log likelihood.

We have extended this to include stratum level grouping, where a stratum includes one or more clusters and weighting to allow finite population correction. We suppose that there are a total of S strata in the sample. Each stratum, 's,' contains C_s clusters. The number of observations in a cluster is N_{cs}. Neglecting any other weighting considerations mentioned below, the full corrected covariance matrix is now

Variance Estimator = **VGV**

V = the inverse of the conventional estimator of the Hessian

$$\mathbf{G} = \sum_{s=1}^{S} w_s \mathbf{G}_s$$

$$\mathbf{G}_s = \left(\sum_{c=1}^{C_s} \mathbf{g}_{cs}\mathbf{g}_{cs}'\right) - \tfrac{1}{C_s}\mathbf{g}_s\mathbf{g}_s'$$

$$\mathbf{g}_{cs} = \sum_{\mathrm{i}=1}^{\mathrm{N}_{cs}} \mathrm{w}_{ics}\mathbf{g}_{ics}$$

$$\mathbf{g}_s = \sum_{\mathrm{c}=1}^{\mathrm{C}_s} \mathbf{g}_{cs}$$

where $\mathbf{g}_{ics}$ is the derivative of the contribution to the log likelihood of individual i in cluster c in stratum s. The remaining detail in the preceding is the weighting factor, w_s. The stratum weight is computed as

$$w_s = f_s \times h_s \times d$$

where $f_s = 1$ or a finite population correction, $1 - C_s/C_s^*$ where C_s^* is the true number of clusters in stratum s, where $C_s^* \geq C_s$.

$h_s = 1$ or $C_s/(C_s - 1)$

$d = 1$ or $(N\text{-}1)/(N\text{-}K)$ where N is the total number of observations in the entire sample and K is the number of parameters (rows in **V**).

Requesting this computation requires use of several switches and specifications in the model command. Use

; Cluster	=	the number of observations in a cluster (fixed) or the name of the identification variable which gives the cluster an identification. This is the setup that is described above.
; Stratum	=	the number of observations in a stratum (fixed) or the name of a stratification variable which gives the stratum an identification.
; Wts	=	the name of the usual weighting variable for model estimation if weights are desired. This defines w_{ics}. This is the usual weighting setup that has been used in all previous versions of *LIMDEP*.
; Fpc	=	the name of a variable which gives the number of clusters in the stratum. This number will be the same for all observations in a stratum – repeated for all clusters in the stratum. If this number is the same for all strata, then just give the number.
; Huber	=	Use this switch to request h_s. If omitted, $h_s = 1$ is used.
; Dfc	=	Use this switch to request the use of d given above. If omitted, $d = 1$ is used.

You may request a summary of the group and stratum sizes to be given after estimation by adding

```
                        ; Describe
```

to the command. Note, **; Describe** produces a line of description for each stratum, so if you have a very large number of strata in your sample, you may want to avoid this option.

This sampling setup may be used with any estimator in *LIMDEP*. One note, however, you should not use it with panel data models. The so called 'clustering' corrections are already built into panel data estimators.

The following shows the setup for a sample that contains 6,350 observations. This is a panel with five observations per individual. We have also artificially divided the sample into five strata, each with 1,270 observations, then fit a probit model. The information below would appear with a model command that used this configuration of the data to construct a robust covariance matrix.

```
PROBIT          ; Lhs = ip ; Rhs = x
                ; Cluster = 5
                ; Stratum = 1270
                ; Describe $
```

These results appear before any results of the probit command. They are produced by the **; Describe** specification in the command.

To continue the example in the previous section, we artificially divided the data set into four levels with

```
CREATE          ; level = 1 + (id > 2000) + (id > 4000) + (id > 6000) $
```

The estimation command that accounts for this second level of grouping is

```
PROBIT          ; Lhs = doctor; Rhs = one,age,educ,married,hsat
                ; Cluster = id  ; Stratum = level ; Describe $
```

```
========================================================================
 Summary of Sample Configuration for Two Level Stratified Data
========================================================================
 Stratum #   Stratum    Number Groups          Group Sizes
            Size (obs)  Sample   FPC.       1       2       3 ...    Mean
==========  ==========  =============  =================================
         1        7617    2000 1.0000       3       4       4 ...     3.8
         2        7963    2000 1.0000       4       1       7 ...     4.0
         3        7796    2000 1.0000       7       6       3 ...     3.9
         4        3950    1292 1.0000       6       1       3 ...     3.1
Normal exit:   4 iterations. Status=0, F=    16639.16
+----------------------------------------------------------------------+
| Covariance matrix for the model is adjusted for data clustering.     |
| Sample of  27326 observations contained   7293 clusters defined by   |
| variable ID       which identifies by a value a cluster ID.          |
| Sample of  27326 observations contained      4 strata defined by     |
| variable LEVEL    which identifies by a value a stratum ID.          |
+----------------------------------------------------------------------+
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               DOCTOR
Log likelihood function    -16639.15794
Restricted log likelihood  -18019.55173
Chi squared [   4 d.f.]      2760.78759
Significance level               .00000
McFadden Pseudo R-squared      .0766053
Estimation based on N =  27326, K =   5
Inf.Cr.AIC  =33288.316 AIC/N =    1.218
Hosmer-Lemeshow chi-squared =  20.38745
P-value=  .00897 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|    1.35413***      .14134     9.58  .0000     1.07711   1.63115
     AGE|     .00849***      .00164     5.18  .0000      .00528    .01170
    EDUC|    -.01544*        .00820    -1.88  .0598     -.03152    .00063
 MARRIED|     .00818         .04168      .20  .8445     -.07352    .08987
HSAT|    -.17506***      .00812   -21.56  .0000     -.19097   -.15914
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

R11: Partial Effects

R11.1 Partial Effects for Estimated Models

After model estimation is completed, the model results will generally be used for three 'post estimation' functions:

- Estimation and analysis of partial effects,
- Prediction, decomposition of predictions, and simulation,
- Hypothesis testing.

This chapter will show how to estimate and analyze partial effects. Predictions and model simulations are detailed in Chapter R12. Hypothesis tests are discussed in Chapter R13. Each of these functions is a general feature of all of the models that you will fit with *LIMDEP*. These three chapters will discuss the overall functions. Aspects of the particular models will be given in the *Econometric Modeling Guide*.

This chapter describes two tools for analyzing partial effects, the **; Partial Effects** specification in model commands and a separate post estimation command **PARTIAL EFFECTS** (or just **PARTIALS**) which is used to analyze the effects in greater detail.

The partial effects in a model are implications of the model, itself. To consider an example, suppose we have fit a binary logit model, which specifies

$$\text{Prob}(y = 1|\mathbf{x}) = \Lambda(\boldsymbol{\beta}'\mathbf{x}); \text{Prob}(y = 0|\mathbf{x}) = \Lambda(-\boldsymbol{\beta}'\mathbf{x})$$

where

$$\Lambda(\boldsymbol{\beta}'\mathbf{x}) = \exp(\boldsymbol{\beta}'\mathbf{x})/[1+\exp(\boldsymbol{\beta}'\mathbf{x})].$$

In particular,

$$\text{Prob}[doctor = 1|age,income] = \Lambda(\beta_1 + \beta_2 age + \beta_3 income)$$

where

$$doctor = 1[\textit{visits to doctor} > 0 \textit{ in observation year}].$$

The estimation step produces estimates of **β**, which are reported by the program as described in Chapter R9. In a nonlinear model such as this one, **β** does not measure the impact of **x** on a feature of the relationship between *y* and **x**. In the logit model, the regression function is

$$E[y|\mathbf{x}] = 0\Lambda(\boldsymbol{\beta}'\mathbf{x}) + 1\Lambda(\boldsymbol{\beta}'\mathbf{x}) = \Lambda(\boldsymbol{\beta}'\mathbf{x}).$$

The effect of changes in **x** on the expected value of *y*|**x** is given by the *partial effect*,

$$\partial E[y|\mathbf{x}]/\partial\begin{pmatrix} Age \\ Income \end{pmatrix} = \Lambda(\boldsymbol{\beta}'\mathbf{x})\Lambda(-\boldsymbol{\beta}'\mathbf{x})\boldsymbol{\beta} = \left[\Lambda'(\boldsymbol{\beta}'\mathbf{x})\right]\times\boldsymbol{\beta} = \Lambda(\boldsymbol{\beta}'\mathbf{x})\Lambda(-\boldsymbol{\beta}'\mathbf{x})\begin{pmatrix} \beta_2 \\ \beta_3 \end{pmatrix},$$

which is estimated separately after estimates of **β** are obtained and it is determined what value of **x** will be used. The partial effects are a multiple of the coefficients.

The model specification

; Partial Effects

is used to request this specific computation. This general specification is discussed in Section R11.5. There are also model specific aspects of this computation discussed in the *Econometric Modeling Guide*.

NOTE: *LIMDEP* Version 9 used the specification **; Marginal Effects** for this request. That usage is still supported, although we now use **; Partial Effects** to be consistent with the **PARTIAL EFFECTS** command discussed in Section R11.4.

Analysis of the partial effects is a useful device for using the model to understand its behavioral implications. Continuing the earlier example, interesting questions that might follow include:

- How does the partial effect of *age* on the probability change as individuals get older?
- Does the effect of changes in *income* on the probability change as *age* increases?
- Is the *income* effect substantively different for women and men?

None of these are revealed by using a simple scaling of the coefficients. These sorts of issues and scenarios are analyzed with the **PARTIAL EFFECTS** command discussed in Section R11.4. The differences between the two ways of requesting partial effects are discussed in Section R11.2. Some modeling and computation issues are discussed in Section R11.3.

R11.2 Command vs. Model Specification

The following shows estimates of the model suggested in Section R11.1:

```
LOGIT          ; Lhs = doctor ; Rhs = one,age,income
               ; Partial Effects $
```

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    -.48962***      .05560    -8.81  .0000     -.59860    -.38064
     AGE|     .02625***      .00113    23.15  .0000      .02403     .02847
  INCOME|    -.31944***      .07091    -4.51  .0000     -.45842    -.18046
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

As noted, partial effects are scaled versions of the coefficients. We can estimate these with **; Partial Effects** added to the command, which produces the results below.

```
-----------------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with
respect to the vector of characteristics
Average partial effects for sample obs.
--------+--------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
  DOCTOR|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     AGE|     .00600***      .00026    23.35  .0000      .00550    .00650
  INCOME|    -.07300***      .01620    -4.51  .0000     -.10475   -.04125
--------+--------------------------------------------------------------------
z, prob values and confidence intervals are given for the partial effect
-----------------------------------------------------------------------------
```

The partial effects for the two variables in the model can also be computed using the command,

PARTIAL EFFECTS ; Effects: age / income $

which produces

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to AGE
Results are computed by average over sample observations
Partial effects for continuous AGE      computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dAGE             Partial    Standard
(Delta method)      Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
APE. Function        .00600     .00025   24.03      .00551      .00649

---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to INCOME
Results are computed by average over sample observations
Partial effects for continuous INCOME   computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dINCOME          Partial    Standard
(Delta method)      Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
APE. Function       -.07300     .01618    4.51     -.10472     -.04128
```

The two approaches produce the same answer. (The very small differences in the confidence intervals arise because **; Partial Effects** uses an analytic expression for the derivatives used for the delta method while **PARTIAL EFFECTS** uses numerical approximations to these derivatives.) To illustrate how larger differences will arise, consider a model with an quadratic term and an interaction term:

$$\text{Prob}[\textit{doctor} = 1|\textit{age},\textit{income},\textit{sex}] = \Lambda(\beta_1 + \beta_2 \textit{age} + \beta_3 \textit{age}^2 + \beta_4 \textit{income} + \beta_5 \textit{sex} + \beta_5 \textit{sex*income})$$

The familiar approach to analysis would be

```
CREATE      ; agesq = age^2 ; sex_incm = sex*income $
LOGIT       ; Lhs = doctor ; Rhs = one,age,agesq,income,sex,sex_incm
            ; Partial Effects $
```

which produces

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|    1.83107***      .20775     8.81  .0000     1.42390   2.23825
     AGE|    -.10148***      .01005   -10.10  .0000     -.12119   -.08178
   AGESQ|     .00145***      .00012    12.61  .0000      .00123    .00168
  INCOME|    -.21588**       .09902    -2.18  .0292     -.40997   -.02180
     SEX|     .50830***      .05774     8.80  .0000      .39513    .62146
SEX_INCM|     .26288*        .14571     1.80  .0712     -.02271    .54846
--------+--------------------------------------------------------------------
Average partial effects for sample obs.
--------+--------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
  DOCTOR|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     AGE|    -.02262***      .00224   -10.11  .0000     -.02700   -.01823
   AGESQ|     .00032***   .2558D-04    12.64  .0000      .00027    .00037
  INCOME|    -.04811**       .02207    -2.18  .0292     -.09136   -.00486
     SEX|     .11459***      .01295     8.85  .0000      .08922    .13996   #
SEX_INCM|     .05858*        .03247     1.80  .0712     -.00506    .12223
--------+--------------------------------------------------------------------
#  Partial effect for dummy variable is E[y|x,d=1] - E[y|x,d=0]
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The processor has produced the scaled coefficients as requested, but these are not the partial effects. In particular, with this specification,

$$\partial \text{Prob}(doctor = 1|\mathbf{x})/\partial age \;=\; \Lambda(\boldsymbol{\beta}'\mathbf{x})\Lambda(-\boldsymbol{\beta}'\mathbf{x})(\beta_2 + 2\beta_3 age).$$

(Note, the problem has arisen because *LIMDEP* does not know from this command that *agesq* is the square of *age*. It could be anything; it is just a name in a list.) It is possible to program the right result, but it is a bit cumbersome. We could proceed as follows:

```
NAMELIST    ; x = one,age,agesq,income,sex,sex_incm $
WALD        ; Labels = b1,b2,b3,b4,b5,b6
            ; Start = b ; Var = varb
            ; Fn1 = age_efct = Lgp(b1’x)*Lgp(-b1’x)*(b2 + 2*b3*age)
            ; Average $
```

```
-----------------------------------------------------------------------------
WALD procedure. Estimates and standard errors
for nonlinear functions and joint test of
nonlinear restrictions.
Wald Statistic             =     53.98486
Prob. from Chi-squared[ 1] =       .00000
Functions of data are averaged over the obs.
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
WaldFcns|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
AGE_EFCT|     .00483***      .00024    20.33  .0000      .00436    .00529
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The **PARTIAL EFFECTS** command is provided to automate this calculation. First, the quadratic term and interaction are built into the command, so *LIMDEP* can find them later. Then, the **PARTIAL EFFECTS** command does the rest.

LOGIT ; Lhs = doctor ; Rhs = one,age,age^2,income,sex,sex*income $

The estimated model is identical, though the estimates are now labeled to show the built in structure.

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|    1.83107***      .20775     8.81  .0000     1.42390   2.23825
     AGE|    -.10148***      .01005   -10.10  .0000     -.12119   -.08178
 AGE^2.0|     .00145***      .00012    12.61  .0000      .00123    .00168
  INCOME|    -.21588**       .09902    -2.18  .0292     -.40997   -.02180
     SEX|     .50830***      .05774     8.80  .0000      .39513    .62146
        |Interaction SEX*INCOME
Intrct02|     .26288*        .14571     1.80  .0712     -.02271    .54846
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The partial effect for *age* is calculated using the analytic result, not as a scaled coefficient.

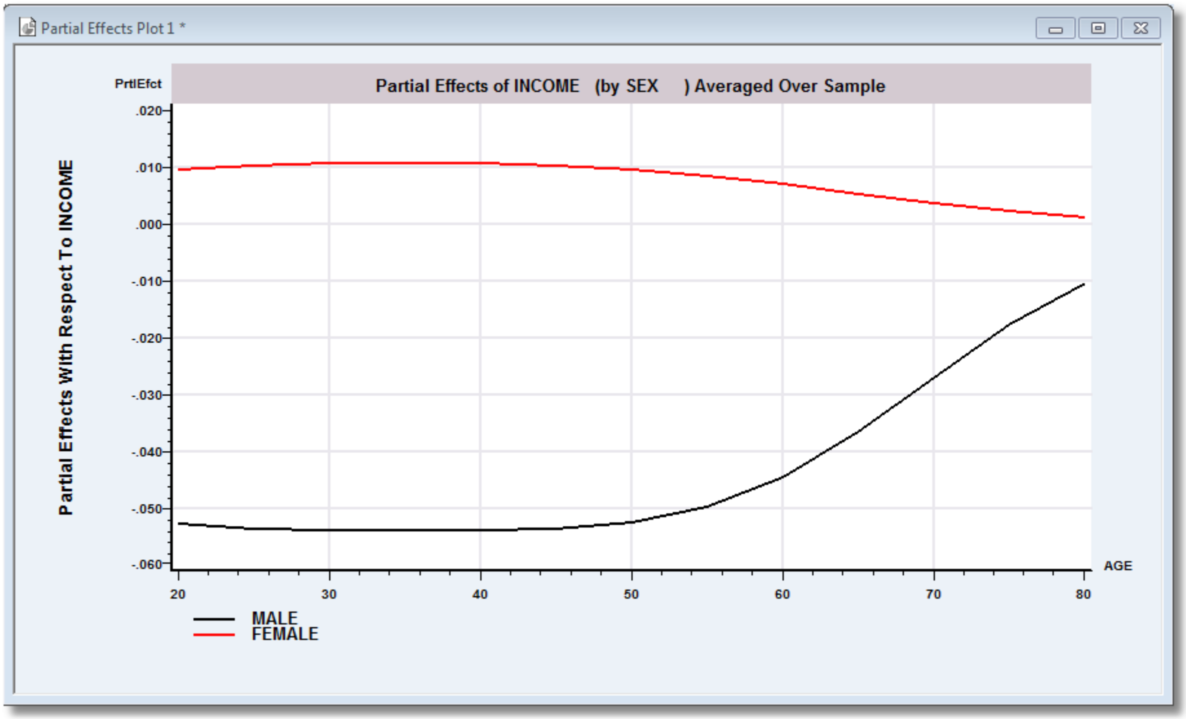
PARTIAL EFFECTS ; Effects: age $

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to AGE
Results are computed by average over sample observations
Partial effects for continuous AGE      computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dAGE              Partial    Standard
(Delta method)       Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
APE. Function         .00483     .00024   20.33      .00436      .00529
```

To this point, **PARTIAL EFFECTS** has merely provided a useful shortcut for obtaining partial effects for a variable when there is a nonlinear term in the model. But, you can go far beyond just automating the partial effects. To consider a final example, 'does the partial effect of *income* on the probability of visiting the doctor vary by sex? And, does it vary systematically by age as well?' The following computes the partial effect of income at ages 20, 25, …, 80, separately for men and women, and plots the two sets of results to reveal the extent and nature of the difference.

```
PARTIAL EFFECTS  ; Effects: income & age = 20(5)80
                                | sex = 0,1 (male,female)
                 ; Plot $
```

```
--------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
--------------------------------------------------------------------
Effects on function with respect to INCOME
Results are computed by average over sample observations
Partial effects for continuous INCOME   computed by differentiation
Effect is computed as derivative     = df(.)/dx
--------------------------------------------------------------------
df/dINCOME         Partial    Standard
(Delta method)     Effect      Error     |t|  95% Confidence Interval
--------------------------------------------------------------------
APE. Function      -.02249     .01642    1.37     -.05468      .00969
--------------------------------------------------------------------
SEX     = MALE     -------------------------------------------------
AGE     = 20.00    -.05271     .02421    2.18     -.10017     -.00525
AGE     = 25.00    -.05374     .02464    2.18     -.10204     -.00544
AGE     = 30.00    -.05395     .02473    2.18     -.10241     -.00548
AGE     = 35.00    -.05395     .02473    2.18     -.10241     -.00548
AGE     = 40.00    -.05395     .02473    2.18     -.10241     -.00548
AGE     = 45.00    -.05374     .02462    2.18     -.10199     -.00548
AGE     = 50.00    -.05269     .02413    2.18     -.09999     -.00539
AGE     = 55.00    -.04990     .02286    2.18     -.09470     -.00510
AGE     = 60.00    -.04455     .02043    2.18     -.08460     -.00449
AGE     = 65.00    -.03656     .01685    2.17     -.06957     -.00354
AGE     = 70.00    -.02702     .01258    2.15     -.05167     -.00236
AGE     = 75.00    -.01780     .00845    2.11     -.03436     -.00124
AGE     = 80.00    -.01046     .00511    2.05     -.02048     -.00044
--------------------------------------------------------------------
SEX     = FEMALE   -------------------------------------------------
--------------------------------------------------------------------
AGE     = 20.00     .00963     .02219     .43     -.03385      .05311
AGE     = 25.00     .01033     .02381     .43     -.03634      .05699
AGE     = 30.00     .01070     .02467     .43     -.03765      .05905
AGE     = 35.00     .01081     .02493     .43     -.03806      .05968
AGE     = 40.00     .01069     .02467     .43     -.03766      .05905
AGE     = 45.00     .01032     .02381     .43     -.03635      .05699
AGE     = 50.00     .00962     .02219     .43     -.03387      .05311
AGE     = 55.00     .00852     .01965     .43     -.03000      .04704
AGE     = 60.00     .00705     .01625     .43     -.02480      .03890
AGE     = 65.00     .00536     .01233     .43     -.01882      .02953
AGE     = 70.00     .00370     .00850     .44     -.01296      .02036
AGE     = 75.00     .00231     .00530     .44     -.00808      .01270
AGE     = 80.00     .00131     .00300     .44     -.00456      .00718
```



There is obviously a great deal more that you can do with the command than with the model specification. To sum up:

Use **; Partial Effects** as a model specification to obtain partial effects in the form of scaled coefficients, in a single convenient table. Other results that this approach will produce are discussed in Section R11.5.

Use **PARTIAL EFFECTS** as a post estimation command for detailed analyses of variables in models, such as the exercise above.

NOTE: When your model includes nonlinear functions and interaction terms built into the specification, as in the example above, then you must use the post estimation command **PARTIAL EFFECTS** to get the appropriate computation of partial effects.

A namelist may appear instead of a variable or a list of variables separated by slashes. When you specify partial effects for a namelist of variables, a separate table is produced for each variable in the list (save for the constant term, which is ignored). It may be useful to have a single table for the set of variables. Use **; Summary** for this purpose. The following example shows how:

```
NAMELIST    ; xdoc = one,age,educ,married $
POISSON     ; Lhs = docvis ; Rhs = xdoc $
PARTIALS    ; Effects : xdoc $
PARTIALS    ; Effects : xdoc ; Summary $
```

The first **PARTIALS** command produces

```
---------------------------------------------------------------------
Partial Effects  Analysis for Exponential Regression Function
---------------------------------------------------------------------
Effects on function with respect to AGE
Results are computed by average over sample observations
Partial effects for continuous AGE      computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dAGE             Partial    Standard
(Delta method)      Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
APE. Function        .06965     .00101   69.00      .06767      .07162
```

(Separate results for *educ* and *married* follow.)

The second produces only a single table.

```
---------------------------------------------------------------------
Partial Effects for Exponential Regression Function
Partial Effects Averaged Over Observations
* ==> Partial Effect for a Binary Variable
---------------------------------------------------------------------
                    Partial    Standard
(Delta method)      Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
      AGE            .06965     .00101   69.00      .06767      .07162
      EDUC         -.17274     .00534   32.33     -.18321     -.16227
   *  MARRIED      -.36333     .02714   13.39     -.41652     -.31013
---------------------------------------------------------------------
```

R11.3 Partial Effects Issues

In single index equation models in which there exists a conditional mean, such as the logit model shown in Section R11.1, the usual choice for a partial effect is the regression function, E[y|$\mathbf{x}$]. The effect is

$$\boldsymbol{\delta} = \partial \text{E[Lhs variable]}/\partial \text{Rhs variables.}$$

This derivative is what one usually has in mind for the partial effect. In a linear regression model, this is simply the vector of regression coefficients. In many other nonlinear models, it will be a vector of scaled coefficients. The Poisson regression model provides a familiar example;

$$E[y|\mathbf{x}] = \exp(\boldsymbol{\beta}'\mathbf{x}) = \lambda(\boldsymbol{\beta}'\mathbf{x}),$$

so that

$$\boldsymbol{\delta} = \lambda(\boldsymbol{\beta}'\mathbf{x})\times\boldsymbol{\beta}.$$

In many models, particularly multiple equation models, there is no obvious choice for what function to analyze. Consider the bivariate probit model, which models the joint probability distribution of two binary outcome variables. To produce 'partial effects,' we must first determine what the margin is. There are at least three candidates, the joint probability, Prob[y_1=1,y_2=1|$\mathbf{x}$], a general conditional mean function, $E[y_1|y_2,\mathbf{x}]$ and a particular conditional mean function, $E[y_1|y_2 = 1,\mathbf{x}]$, and there are others. For another example, in an ordered probit or ordered logit model, the specification provides

$$\text{Prob}[y = j|\mathbf{x}] \;=\; H_j(\boldsymbol{\beta},\boldsymbol{\mu}.\mathbf{x}),\; j = 0,1,\ldots,J.$$

The health satisfaction model analyzed in Greene and Hensher (2010) involves a response variable that takes values 0,1,…,10, eleven values. So, there are eleven different probability functions that can be differentiated and, more to the point, there is no regression function. (An example with three outcomes appears in Section R11.5.1.) In general, *LIMDEP* does provide an answer for such models, which will be fully documented, but we emphasize, it might not be precisely what you are looking for, and you may have to do some additional computation to get the precise result you seek. The **PARTIAL EFFECTS** command described in Section R11.4 will allow you to obtain partial effects for any function of interest.

There is a long list of issues that you want to be aware of in computing, reporting and analyzing partial effects: For an example, in the logit model suggested earlier,

$$\begin{aligned}&\text{Prob}[\mathit{doctor} = 1|\mathit{age},\mathit{sex},\mathit{income}]\\ &\qquad = \Lambda(\beta_1 + \beta_2 \mathit{age} + \beta_3 \mathit{age}^2 + \beta_4 \mathit{income} + \beta_5 \mathit{sex} + \beta_6\, \mathit{sex}\times\mathit{income})\\ &\qquad = \Lambda(\boldsymbol{\beta}'\mathbf{x}),\end{aligned}$$

three partial effects are

$$\begin{aligned}\Delta\text{Prob}[\mathit{doctor} = 1|\mathbf{x}]/\Delta \mathit{sex} &= \Lambda(\boldsymbol{\beta}'\mathbf{x}|\mathit{sex}=1) - \Lambda(\boldsymbol{\beta}'\mathbf{x}|\mathit{sex}=0)\\ \partial\text{Prob}[\mathit{doctor} = 1|\mathbf{x}]/\partial \mathit{age} &= \Lambda(\boldsymbol{\beta}'\mathbf{x})\Lambda(-\boldsymbol{\beta}'\mathbf{x}) \times (\beta_2 + 2\beta_3 \mathit{age})\\ \partial\text{Prob}[\mathit{doctor} = 1|\mathbf{x}]/\partial \mathit{income} &= \Lambda(\boldsymbol{\beta}'\mathbf{x})\Lambda(-\boldsymbol{\beta}'\mathbf{x}) \times (\beta_5 + \beta_6 \mathit{sex})\end{aligned}$$

The following example shows the typical approach to estimation of this model, with data transformation and partial effects, using the health care data used in the earlier examples:

```
CREATE    ; agesq = age*age ; sex_incm = sex*income $
LOGIT     ; Lhs = doctor
          ; Rhs = one,age,agesq,income, sex,sex_incm
          ; Partial Effects $
```

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|    1.83107***      .20775     8.81  .0000     1.42390   2.23825
     AGE|    -.10148***      .01005   -10.10  .0000     -.12119   -.08178
   AGESQ|     .00145***      .00012    12.61  .0000      .00123    .00168
  INCOME|    -.21588**       .09902    -2.18  .0292     -.40997   -.02180
     SEX|     .50830***      .05774     8.80  .0000      .39513    .62146
SEX_INCM|     .26288*        .14571     1.80  .0712     -.02271    .54846
--------+--------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with respect to the vector of
characteristics. Average partial effects for sample obs.
--------+--------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
  DOCTOR|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     AGE|    -.02262***      .00224   -10.11  .0000     -.02700   -.01823
   AGESQ|     .00032***   .2558D-04    12.64  .0000      .00027    .00037
  INCOME|    -.04811**       .02207    -2.18  .0292     -.09136   -.00486
     SEX|     .11459***      .01295     8.85  .0000      .08922    .13996   #
SEX_INCM|     .05858*        .03247     1.80  .0712     -.00506    .12223
--------+--------------------------------------------------------------------
#  Partial effect for dummy variable is E[y|x,d=1] - E[y|x,d=0]
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

In obtaining partial effects, the following are issues to be considered:

1. What is the function that to be analyzed? There is no single ‘right’ answer to this question. Researchers are usually interested in the slopes of the conditional mean function. But, some other function might be of interest. And, in many models, such as the ordered probit and multinomial logit models, there is no conditional mean function. In the example above, we are analyzing the probability, or the conditional mean function.

2. Partial effects can be computed at the means of the variables, some other specific values of the variables, or averaged over the sample observations on the variables. As noted in the footnotes to the first table, the example above provides partial effects averaged over the sample observations. This is more or less the norm in the recent literature.

3. Partial effects for binary variables such as *sex* should generally be computed using first differences, as shown above, not by scaling coefficients. The results above indicate that the program has noticed that *sex* is a dummy variable, and adjusted the computations accordingly.

4. Qualitative variables can produce an ambiguity. For example, if *schooling* in a model is coded using two dummy variables representing *high school* (base case, $E1 = 0$, $E2 = 0$), *trade school or college* ($E1 = 1$, $E2 = 0$) and *post graduate* ($E1 = 0$, $E2 = 1$), then how should the 'effect' of the third category, $E2$, be measured? Treating $E2$ as a dummy variable the same as *sex* compares *post graduate* education to *high school*. But, since one normally attends college before going to graduate school, the interesting comparison might be between $E2$ and $E1$. This can be handled by manipulating the partial effects, or by recoding $E2$ as $E2' = E1+E2$. But, either way, one wants to maintain the distinction.

5. Compound models such as two part models or heteroscedasticity models may have variables that appear at more than one place in the equation. For example, the conditional mean function in the zero inflated Poisson regression model is of the form $F(\boldsymbol{\gamma}'\mathbf{z})\times\lambda(\boldsymbol{\beta}'\mathbf{x})/[1\text{-exp(-}\lambda(\boldsymbol{\beta}'\mathbf{x})]$. Not only does this model contain two sets of covariates to analyze, **z** and **x**, but in most cases, some variables would appear in both **z** and **x**. Partial effects involve variables in both parts of the model. The partial effect of a common variable, *w*, would be the sum of the two parts. It is worth noting, this compound effect might be quite different from the coefficients, in sign and magnitude.

6. In the model above that contains both *age* and age^2, the partial effect is not a simple scaling of the coefficient. In general, a partial effects calculation that reports a scaled coefficient vector for an index function model will not do this calculation correctly – it will incorrectly compute separate effects for '*age*' and 'age^2.' The results in the preceding example are not correct – the partial effect for *age* is not -0.02206 and the effect reported for *agesq* makes no sense. The problem is that the logit estimator has no way to know that *agesq* is the square of *age*; they are just two variables in the model. (The **PARTIAL EFFECTS** procedure described below solves this problem.)

7. The partial effects for *sex* and *income* in the model above are determined partly by the interaction term. This is not computed correctly by a scaled coefficient. The results above do not make the connection, again because the program has no way to know that the variable named *sex_incm* is equal to *sex* times *income*.

8. The interaction term, itself, creates an ambiguity in the interpretation of the model. What is the 'interaction effect?'

9. Researchers differ on whether standard errors and hypothesis tests about partial effects should be computed.

LIMDEP provides two settings for computing partial effects, the **; Partial Effects** calculations built into the model commands and a separate program, the **PARTIAL EFFECTS** command.

Some of the problems listed above are handled automatically by **; Partial Effects** when you estimate the model. In general, **; Partial Effects** produces a convenient table of scaled coefficients for all of the variables in the model at the same time. The results are appropriate for models that do not contain interaction or nonlinear terms and when the set of scaled coefficients is your desired result. For the items listed above, **; Partial Effects** works as follows:

1. A specific choice, usually the conditional mean function or the probability in the model is analyzed.
2. The usual calculation is the average partial effect. The **; Means** option is provided to request the calculation at the means.
3. Binary variables are usually automatically detected.
4. Qualitative variables are not handled directly.
5. Compound models are generally handled automatically.
6. Nonlinearities in the variables are generally not handled correctly.
7. Interaction effects are not handled correctly.
8. Interaction effects must be analyzed separately.
9. Standard errors and confidence intervals are provided with the estimates.
10. Nearly all models, including all panel data models, display elasticities with partial effects.

All of these issues are handled directly and completely by the **PARTIAL EFFECTS** command. You will use **PARTIAL EFFECTS** to do detailed analysis of a particular variable, rather than all variables simultaneously. The **PARTIAL EFFECTS** command provides the following:

1. There is a default function assumed for each model, but you can specify a different function to be analyzed.
2. Effects can be averaged across observations, computed at the means, or at specified values of particular variables.
3. Binary variables are always handled appropriately.
4. The switch between categories for a categorical variable can be built into the calculation.
5. Effects in compound models are automatically accounted for.
6. All nonlinearities are accounted for in computed effects.
7. All interaction effects are accounted for in computed effects.
8. Effects and interaction terms can be analyzed numerically or graphically.
9. Standard errors and confidence intervals are provided for all computed effects.

You should use **; Partial Effects** for simple index function models such as a basic probit model with a simple specification. You should use **PARTIAL EFFECTS** for more involved model specifications and to analyze in more detail the implications of your model for the interactions of all of its components.

R11.4 The PARTIAL EFFECTS Command

The **PARTIAL EFFECTS** (or just **PARTIALS**) command is structured to work with the new model syntax described in Section R8.3. An example will help to fix ideas. The logit model examined in Section R11.3 was specified using

```
CREATE    ; agesq = age*age ; sex_incm = sex*income $
LOGIT     ; Lhs = doctor
          ; Rhs = one,age,agesq,sex,income,sex_incm
          ; Partial Effects $
```

The same model can be specified by building the quadratic and interaction terms into the equation, as

```
LOGIT     ; Lhs = doctor
          ; Rhs = one,age,age^2,sex,income,sex*income
          ; Partial Effects $
```

The estimation results produced by this alternative command are the same as the first, though the labeling is slightly different – the program notices the quadratic and interaction terms:

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|     2.06138***      .52067     3.96  .0001     1.04089   3.08187
     AGE|     -.10040***      .02531    -3.97  .0001     -.15001   -.05079
 AGE^2.0|      .00141***      .00029     4.88  .0000      .00085    .00198
     SEX|      .46495***      .15131     3.07  .0021      .16839    .76150
  INCOME|     -.49829*        .26152    -1.91  .0567    -1.01086    .01429
        |Interaction SEX*INCOME
Intrct02|      .21012         .38785      .54  .5880     -.55005    .97028
--------+--------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with
respect to the vector of characteristics
Average partial effects for sample obs.
--------+--------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
  DOCTOR|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     AGE|     -.02206***     -.07649    -3.97  .0001     -.03295   -.01118
 AGE^2.0|      .00031***      .03798     4.89  .0000      .00019    .00044
     SEX|      .10302***      .00142     3.09  .0020      .03766    .16838    #
  INCOME|     -.10950*       -.00421    -1.91  .0567     -.22211    .00312
        |Interaction SEX*INCOME
Intrct02|      .04617         .00032      .54  .5880     -.12087    .21322
--------+--------------------------------------------------------------------
```

As we noted earlier, none of the partial effects in this table are actually correct. Regardless of the model specification, **; Partial Effects** reports the scaled coefficient vector. The *age* effect does not account for the quadratic term, the *sex* effect does not account for the interaction with *income* and the *income* effect does not account for its interaction with *sex*. The basic post estimation command

PARTIAL EFFECTS ; Effects: income / sex / age $

produces the following three sets of results:

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit:Probability(DOCTOR=1)
---------------------------------------------------------------------
Results are computed by average over sample observations
Partial effects for continuous INCOME   computed by differentiation
---------------------------------------------------------------------
df/dINCOME         Partial    Standard
(Delta method)     Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
APE. Function      -.04811     .02206    2.18     -.09135     -.00487
---------------------------------------------------------------------
Partial effects for binary var SEX      computed by first difference
---------------------------------------------------------------------
APE. Function       .11459     .01295    8.85      .08922      .13996
---------------------------------------------------------------------
Partial effects for continuous AGE      computed by differentiation
---------------------------------------------------------------------
APE. Function      -.02262     .00223   10.16     -.02698     -.01825
---------------------------------------------------------------------
```

All of the effects built into the model command are accounted for in the partial effects. These are the correct estimates of the average partial effects for these three variables in this model.

NOTE: The list of variables in the **PARTIAL EFFECTS** command may be in a namelist. The following produce the same results for the preceding example:

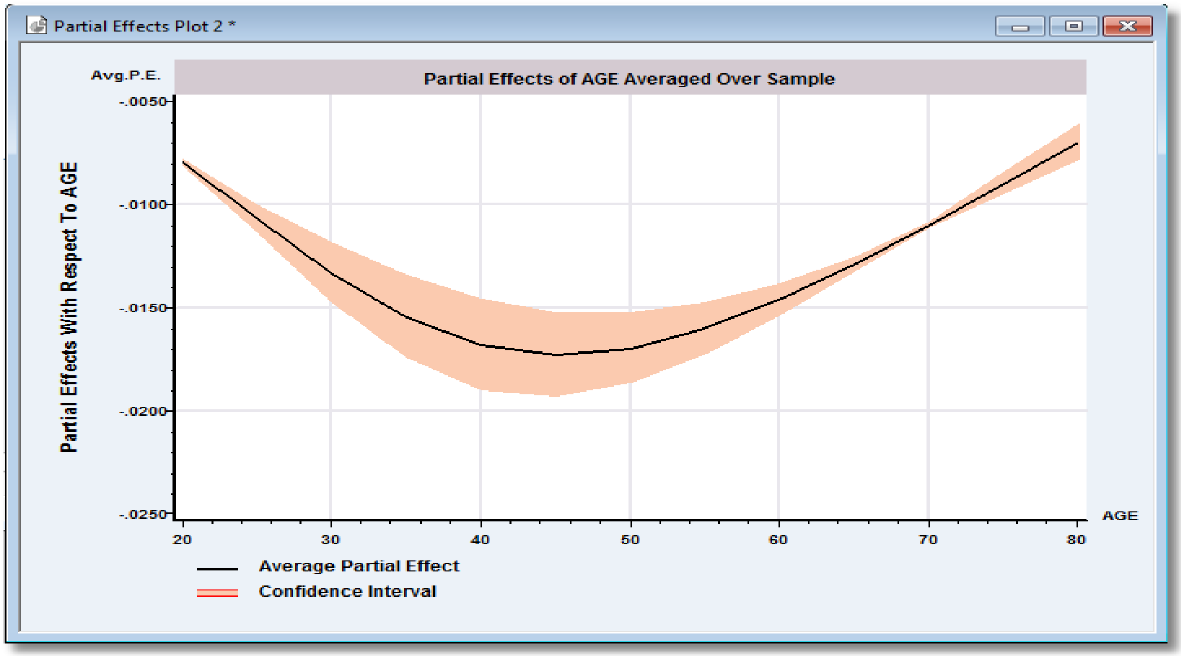
```
NAMELIST            ; x = income, sex, age $
PARTIAL EFFECTS     ; Effects: x $
```

If the namelist contains one (a constant term), the constant term will be ignored in the table of results. I.e., no partial effects are computed for one.

The descriptions of the results are more detailed in this case. There are several options available for changing the computation. Sections R11.4.1-R11.4.6 describe in detail the specifications and options used with the **PARTIAL EFFECTS** command. To suggest how this new feature extends the reach of your model analysis, here is a more detailed example. The first table above shows that the average partial effect of *age* on the probability of visiting the doctor is 0.00428, averaged over the sample. But, the model contains a nonlinearity in *age*. The partial effect with respect to *age* varies with *age*, both because the probability is a nonlinear function and because of this quadratic term. We can analyze this in more detail as follows:

PARTIAL EFFECTS ; Effects: age & age = 20(5)80 ; Plot(ci) $

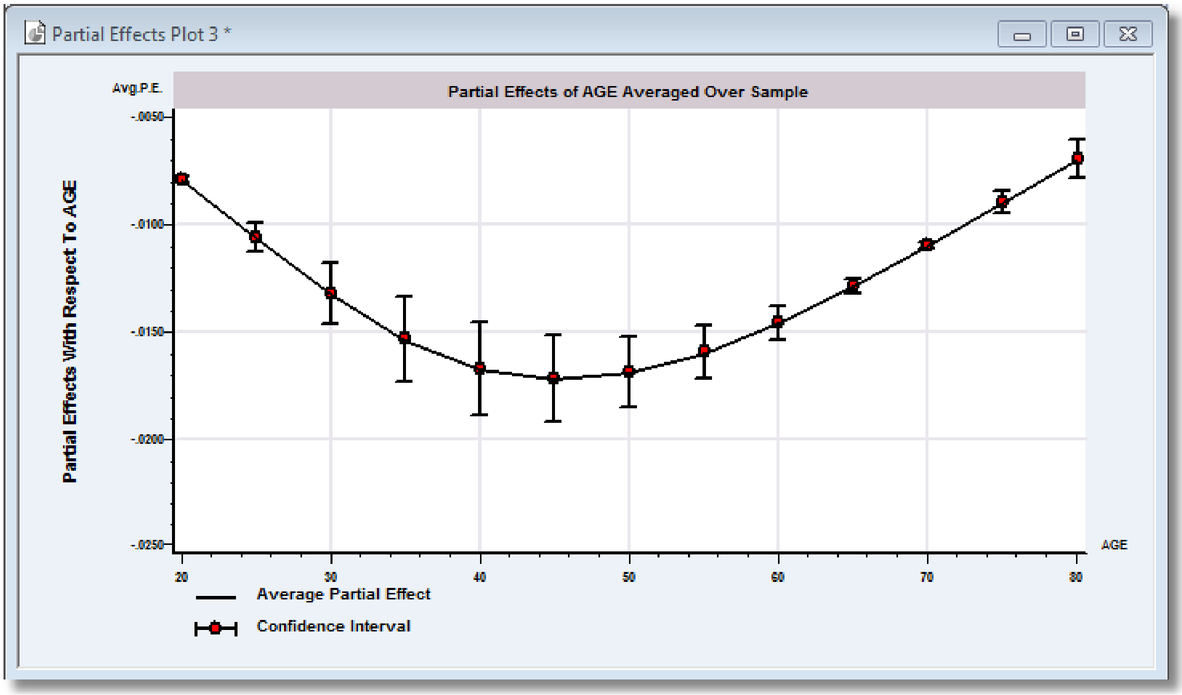
```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to AGE
Results are computed by average over sample observations
Partial effects for continuous AGE      computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dAGE              Partial    Standard
(Delta method)       Effect      Error      |t|  95% Confidence Interval
---------------------------------------------------------------------
Partial effect        .00428     .00058    7.35       .00314       .00542
APE. Function       -.02262     .00223   10.16      -.02698      -.01825
AGE      = 20.00    -.00798     .00009   91.71      -.00815      -.00781
AGE      = 25.00    -.01067     .00034   31.48      -.01133      -.01000
AGE      = 30.00    -.01330     .00073   18.22      -.01473      -.01187
AGE      = 35.00    -.01545     .00101   15.26      -.01743      -.01346
AGE      = 40.00    -.01680     .00111   15.16      -.01897      -.01463
AGE      = 45.00    -.01727     .00103   16.72      -.01930      -.01525
AGE      = 50.00    -.01696     .00085   19.88      -.01863      -.01529
AGE      = 55.00    -.01602     .00063   25.40      -.01726      -.01478
AGE      = 60.00    -.01463     .00040   36.34      -.01542      -.01384
AGE      = 65.00    -.01292     .00018   70.97      -.01328      -.01257
AGE      = 70.00    -.01101     .00007  156.35      -.01115      -.01087
AGE      = 75.00    -.00898     .00027   33.83      -.00951      -.00846
AGE      = 80.00    -.00699     .00044   15.72      -.00786      -.00611
```



Note that the partial effect varies with age and is smallest at about 45. Thus, the aging process implied is that the probability of doctor visitation decreases with age until (individual in this sample) age about 45, then it begins to increase. Although one might suspect this from the different signs of the linear and quadratic terms, the more detailed description is not obvious from the numbers alone.

You can change the style of the confidence limits in the figure by changing ‘ci’ in the **; Plot** instruction to ‘c’ as in the following:

PARTIAL EFFECTS ; Effects: age & age = 20(5)80 ; Plot(c) $



R11.4.1 Last Model Used for Partial Effects

The model used for the **PARTIAL EFFECTS** operation is the last one that you estimated. This will be obvious from the results in your output window, though it is necessary to be specific about which function is being used. That is, what function is being used to compute the effects. For example, the preceding examples are based on a logit model, fit with

```
LOGIT        ; Lhs = doctor
             ; Rhs = one,age,age^2,sex,income,sex*income $
```

The function used for the partial effects is the logit probability,

Last model = $\Lambda(\beta'x)$ = Prob(Lhs variable = 1).

There is a specific function used for each model for which you can use **PARTIAL EFFECTS**. These are documented in the *Econometric Modeling Guide* for the particular models. At any time, you can find out what function is being used for the **PARTIAL EFFECTS** command by using the command

```
LAST MODEL $
```

For our logit example, the response would be

```
--> LAST MODEL $
The last estimated model is Logit Probability Function
```

In most cases, the function used is the conditional mean function. But, in some cases, such as the ordered probit or logit models, there are numerous probability functions. For this particular case, the default function is the probability of the highest outcome, for example,

```
--> OPROBIT ; Lhs = hsat ; Rhs=one,age,educ $
--> LAST MODEL $
The last estimated model is Ordered Probit Probability Y = 10
```

The ordered probit models are a special case. The highest category is usually the one of interest. You can change this by using

; Outcome = value (0,1,…)

NOTE: There is a default function for each model that **PARTIAL EFFECTS** may be used with. However, you can specify a different function to be analyzed. The alternative function need not even be a model. It can be any function you can specify with the command language. **PARTIAL EFFECTS** can analyze any variable in any function that is computed using data and parameters. Section R11.4.6 describes how to supply your own function to be analyzed.

R11.4.2 Sample Used for PARTIAL EFFECTS

The observations used to compute partial effects are whatever happen to be in the current sample. These need not be the observations used to compute the model. The current sample can be a subset of the estimation sample, a completely different set of observations, or even a single observation. The computation of the partial effects is not dependent on the sample used for the estimation.

R11.4.3 Types of Variables in Partial Effects

The central part of the **PARTIAL EFFECTS** command is the request itself and the variable that is changing. (The function to be analyzed is supplied by the previous model command or a specification described in Section R11.4.6.) The simplest form would be

PARTIAL EFFECTS ; Effects: X variable $

This specification requests analysis of

$$\delta_X = \partial f(\text{ last model function }) / \partial X$$

For example, in the logit model estimated earlier, the partial effect for *age* is obtained below.

```
--> PARTIAL EFFECTS ; Effects: age $
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to AGE
Results are computed by average over sample observations
Partial effects for continuous AGE      computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dAGE              Partial    Standard
(Delta method)       Effect      Error      |t|  95% Confidence Interval
---------------------------------------------------------------------
Partial effect        .00428      .00058    7.35      .00314      .00542
```

- The X variable, *age* is noted.
- The processor detects if the X variable is a dummy variable and changes the computation accordingly.

In the results below, the partial effect with respect to *sex* is reported.

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to SEX
Results are computed by average over sample observations
Partial effects for binary var SEX      computed by first difference ←
---------------------------------------------------------------------
df/dSEX              Partial    Standard
(Delta method)       Effect      Error      |t|  95% Confidence Interval
---------------------------------------------------------------------
Partial effect        .11932      .01403    8.50      .09183      .14682
```

R11.4.4 Types of Partial Effects

The default calculation in partial effects is a derivative,

$$\delta_X = \partial f(\text{ last model function }) / \partial X$$

But, there are other functions that might be of interest. You may specify any of the following:

- Elasticity df(X)/dX Use **; Effects: [X variable]**

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to INCOME
Results are computed by average over sample observations
Partial effects for continuous INCOME   computed by differentiation
Effect is computed as elasticity     = dlnf(.)/dlnx ←
---------------------------------------------------------------------
df/dINCOME           Partial    Standard
(Delta method)       Effect      Error      |t|  95% Confidence Interval
---------------------------------------------------------------------
Partial effect       -.05294      .02559    2.07     -.10310     -.00278
```

- Semielasticity dlnF(X)/dX Use **; Effects: < X variable >**

- Partial elasticity dF(X)/dlnX Use **; Effects: { X variable }**

The semielasticity might be used for a conditional mean function for a continuous variable and a discrete X variable such as time. The partial elasticity might be used for a discrete variable and a continuous regressor. For example, the function might be a predictor of the number of visits to the hospital or doctor, or recreation site, etc. while the regressor might be something like income. Finally, if you have a set of categories in your model, you can specify that the margin be the switch from one category to another. The specification requires both variables,

- Category switch f(Cat. A = 1, Cat. B = 0) **-** f(Cat. A = 0, Cat. B = 1).

For example, to compare two regions of the country, you might use

; Effects: Midlands, Cotswalds

Where *midlands* and *cotswalds* are two (among a set of) dummy variables that indicate region of the country. Finally, there is special code used to deal with an ambiguous case. Suppose X is a dummy variable for which you wish to compute the partial effect. If you are using **; Means**, it is not possible to see from the data that the variable is binary. To cover this case, use

- Dummy variable mean f(X = 1) – f(X = 0) Use **; Effects: (variable)**

The two category case is handled likewise with (X_A, X_B).

- Arc elasticity for a dummy variable = proportional effect Use **; Effects: <variable>**

For example, using **; Effects: <sex>** in the preceding model produces

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit:Probability(DOCTOR=1)
---------------------------------------------------------------------
Effects on function with respect to SEX
Results are computed by average over sample observations
Partial effects for binary var SEX      computed by first difference
Arc elasticity is computed for effects of a dummy variable.
[F(x|d=1)-F(x|d=0)] / {(1/2)[F(x|d=1)+F(x|d=0)]}
---------------------------------------------------------------------
df/dSEX              Partial    Standard
(Delta method)       Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
APE. Function        .18551     .02081    8.91      .14472      .22630
```

R11.4.5 Scenarios in the PARTIAL EFFECTS Command

The basic syntax for the **PARTIAL EFFECTS** command is

```
PARTIAL EFFECTS  ; Effects:  variable …  scenario  /
                             variable …  scenario / … $
```

You may provide a scenario for each variable specified. The variable(s) in the scenario can be the same from one to the next or different. The scenario, itself, is optional. The simplest form of the command would be

```
Model            ; Lhs = … ; Rhs = … ; other specifications $
PARTIAL EFFECTS  ; Effects:  an X variable that appears in the model $
```

To obtain a set of partial effects for the variables in a model, separate the names with slashes. For example,

```
LOGIT            ; Lhs = … ; Rhs = one,age,income,sex $
PARTIAL EFFECTS  ; Effects:  age / income / sex $
```

The scenarios are specified as follows:

Discrete Values of a Variable: The '|'Specification

This specification computes the partial effect of the X variable while setting the Z variable equal to the specified values for every observation. The partial effects are computed for each value of Z specified.

X variable | Z variable = value, value, value … up to 10 values

The Z variable may be the same as the X variable or a different variable. For example,

Effects: income | educ = 12, 16, 20

The variable that is changing may be the one that is being analyzed, as in

Effects: educ | educ = 12, 16, 20

You may provide labels in parentheses for the values, as in

```
Effects: income | sex = 0, 1 (male,female)
                / educ = 12,16,20 (hs,college,graduate)
```

The values of the Z variable can be any specified values. They need not be values that occur in the sample. For example, even though *sex* is coded 0,1 in the sample, you could specify '**| sex = 0,1,2,3**'

Range of Values in Steps: The '&' Specification

This specification computes the partial effects for the sequence of

X variable & Z variable = lower limit (step length) upper limit.

For example, as in the application above,

Effects: income & age = 20(5)80

Combining Scenarios

You can combine the two types of scenarios in a single analysis. The general form of the scenario would be

Effects: X variable | Z variable = z1, z2, … & W variable = lower(delta) upper

In a compound scenario, the W variable changes inside the Z variable. That is, the values of W are computed for each value of Z in turn. In the example below, we have *sex* = 0,1 and *age* = 20(5)80. The string of effects is computed for (*sex* = 0 (*male*), age = 20, 25,…,80) then for (*sex* = 1 (*female*), age = 20, 25, …, 80). The command and results are

PARTIAL EFFECTS ; Effects: income & age = 20(5)80
| sex = 0,1 (male,female) $

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to INCOME
Results are computed by average over sample observations
Partial effects for continuous INCOME   computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dINCOME          Partial    Standard
(Delta method)      Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
APE. Function       -.04811     .02206    2.18     -.09135     -.00487
---------------------------------------------------------------------
SEX     = MALE     ---------------------------------------------------
AGE     = 20.00    -.01946     .00920    2.11     -.03749     -.00142
AGE     = 25.00    -.02548     .01183    2.15     -.04867     -.00229
AGE     = 30.00    -.03097     .01428    2.17     -.05897     -.00297
AGE     = 35.00    -.03498     .01614    2.17     -.06661     -.00334
AGE     = 40.00    -.03701     .01716    2.16     -.07064     -.00339
AGE     = 45.00    -.03713     .01731    2.15     -.07106     -.00320
AGE     = 50.00    -.03570     .01675    2.13     -.06852     -.00287
AGE     = 55.00    -.03311     .01564    2.12     -.06377     -.00245
AGE     = 60.00    -.02971     .01415    2.10     -.05744     -.00197
AGE     = 65.00    -.02573     .01239    2.08     -.05001     -.00145
AGE     = 70.00    -.02139     .01047    2.04     -.04191     -.00088
AGE     = 75.00    -.01695     .00850    2.00     -.03360     -.00030
AGE     = 80.00    -.01272     .00661    1.93     -.02567      .00023
---------------------------------------------------------------------
SEX     = FEMALE   ---------------------------------------------------
AGE     = 20.00    -.01388     .00693    2.00     -.02746     -.00030
AGE     = 25.00    -.01945     .00942    2.06     -.03791     -.00098
AGE     = 30.00    -.02547     .01209    2.11     -.04917     -.00177
AGE     = 35.00    -.03096     .01452    2.13     -.05942     -.00251
AGE     = 40.00    -.03497     .01629    2.15     -.06690     -.00304
AGE     = 45.00    -.03701     .01720    2.15     -.07073     -.00329
AGE     = 50.00    -.03714     .01728    2.15     -.07100     -.00327
AGE     = 55.00    -.03570     .01666    2.14     -.06835     -.00305
AGE     = 60.00    -.03311     .01552    2.13     -.06353     -.00270
AGE     = 65.00    -.02971     .01401    2.12     -.05716     -.00226
AGE     = 70.00    -.02574     .01224    2.10     -.04974     -.00174
AGE     = 75.00    -.02140     .01033    2.07     -.04166     -.00115
AGE     = 80.00    -.01696     .00839    2.02     -.03341     -.00050
```

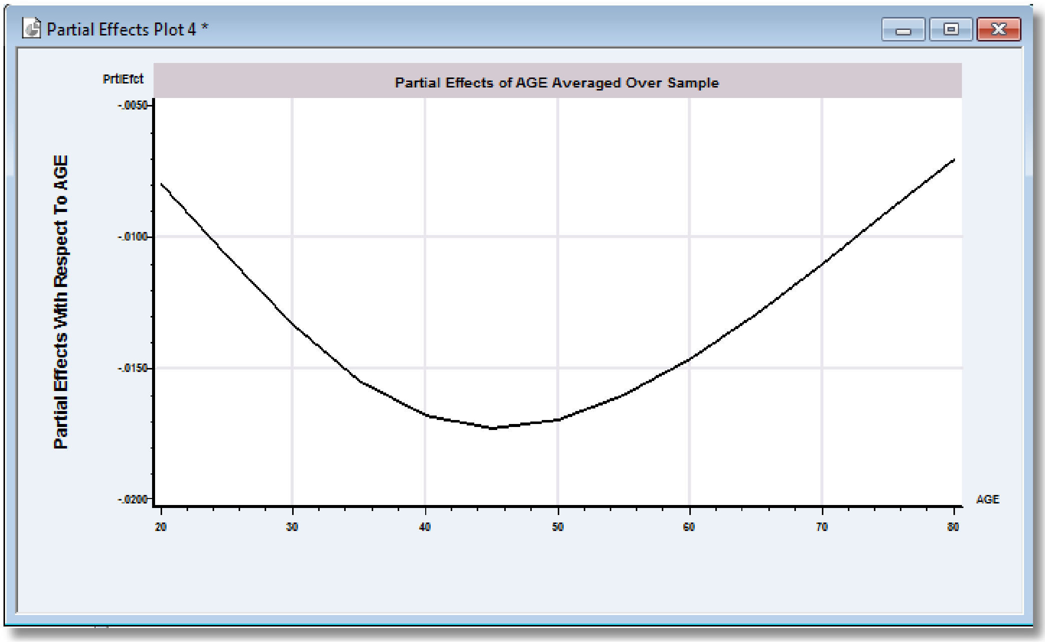
R11.4.6 Plotting Partial Effects

You can produce two types of plots with **PARTIAL EFFECTS**. When the figure has a set of values for a single scenario, for example,

PARTIAL EFFECTS ; Effects: age & age = 20(5)80 ; Plot $

You can request a plot of the partial effects against the specified values of the Z variable. For our logit model, the command produces

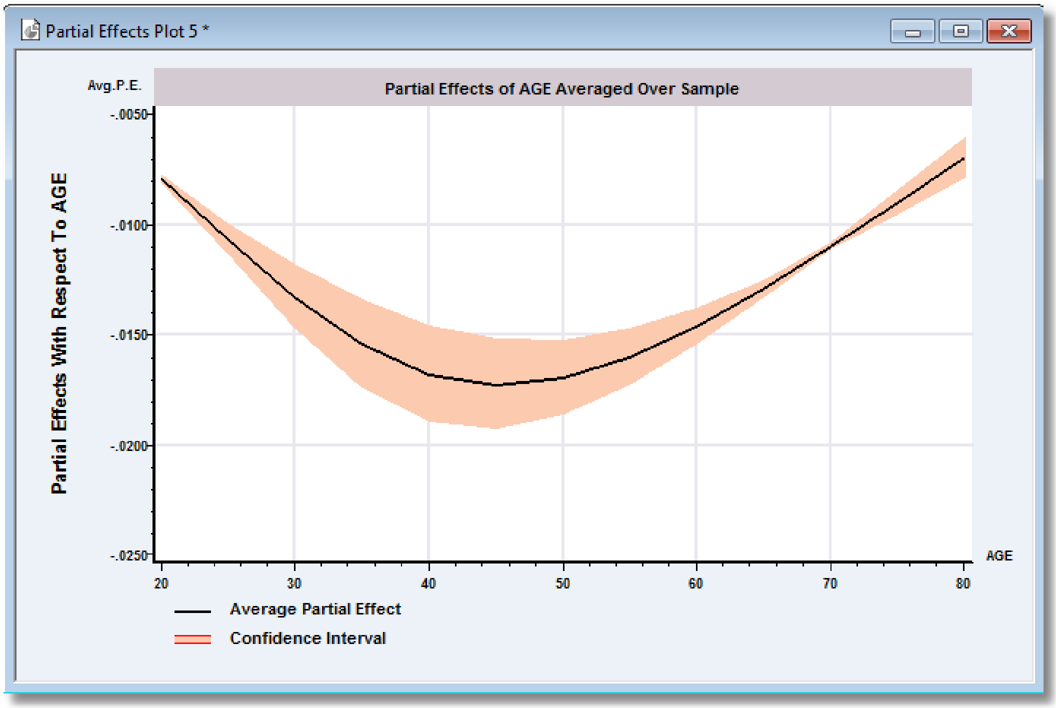
```
-----------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
-----------------------------------------------------------------------
df/dINCOME          Partial    Standard
(Delta method)      Effect      Error      |t|  95% Confidence Interval
-----------------------------------------------------------------------
APE. Function       -.02262     .00223    10.16     -.02698      -.01825
AGE      = 20.00    -.00798     .00009    91.71     -.00815      -.00781
AGE      = 25.00    -.01067     .00034    31.48     -.01133      -.01000
AGE      = 30.00    -.01330     .00073    18.22     -.01473      -.01187
AGE      = 35.00    -.01545     .00101    15.26     -.01743      -.01346
AGE      = 40.00    -.01680     .00111    15.16     -.01897      -.01463
AGE      = 45.00    -.01727     .00103    16.72     -.01930      -.01525
AGE      = 50.00    -.01696     .00085    19.88     -.01863      -.01529
AGE      = 55.00    -.01602     .00063    25.40     -.01726      -.01478
AGE      = 60.00    -.01463     .00040    36.34     -.01542      -.01384
AGE      = 65.00    -.01292     .00018    70.97     -.01328      -.01257
AGE      = 70.00    -.01101     .00007   156.35     -.01115      -.01087
AGE      = 75.00    -.00898     .00027    33.83     -.00951      -.00846
AGE      = 80.00    -.00699     .00044    15.72     -.00786      -.00611
```



You can also add confidence bounds to the figure by changing **; Plot** to **; Plot(ci)**

; Plot(ci) for confidence interval

For this example, we obtain:



The confidence limits are those shown in the table of results. These limits are based on the estimate of the standard error of the average partial effect. In particular, the average partial effect is

$$\delta_X = \frac{1}{N}\sum_{i=1}^{N} \delta_{X,i}\left(\hat{\boldsymbol{\beta}},\mathbf{x}_i\right)$$

The standard error for the estimated δ_K is computed using the delta method,

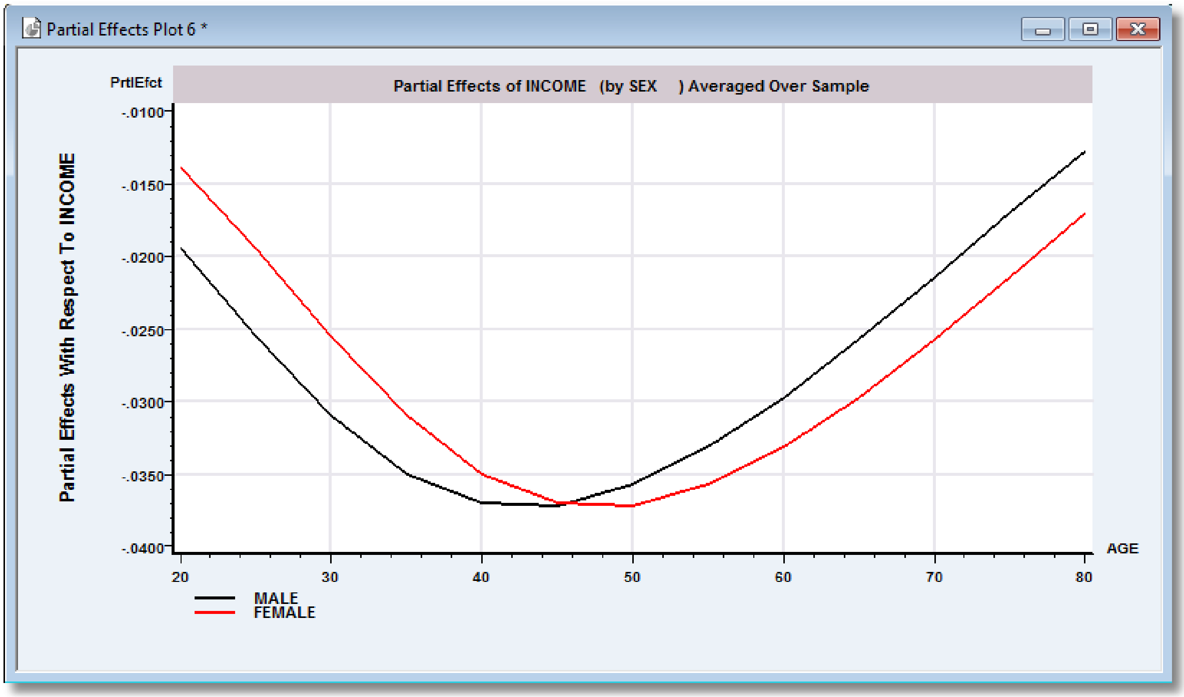
$$V_X = \left[\frac{1}{N}\sum_{i=1}^{N}\frac{\partial\delta_{X,i}\left(\hat{\boldsymbol{\beta}},\mathbf{x}_i\right)}{\partial\hat{\boldsymbol{\beta}}'}\right]\hat{\boldsymbol{\Sigma}}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{\partial\delta_{X,i}\left(\hat{\boldsymbol{\beta}},\mathbf{x}_i\right)}{\partial\boldsymbol{\beta}}\right].$$

The partial effect, δ_X and the square root of V_X is computed for each value Z . Those are the values in the table above. You can decorate the figure a bit by changing the title at the top and the label for the vertical axis with

; Title = title for the figure
; Vaxis = descriptor for the vertical axis

A second type of figure can be produced when you combine & Z variable with | W variable. In this case, up to five plots can appear in the same figure. The example below compares the income effects for men and women.

```
PARTIAL EFFECTS  ; Effects:income
                           | sex = 0,1 (male,female)
                           &age = 20(5)80
                 ; Plot $
PARTIAL EFFECTS  ; Effects:income
                           | sex = 0,1 (male,female)
                           &age = 20(5)80
                 ; Plot $
```



R11.4.7 Sample Partitioning: The '@' Specification

Up to this point, we have used the entire current sample in computing the partial effects, either in computing the average partial effects or in computing the sample means. You can partition the current sample with the following syntax:

X **variable @ F variable**

X **variable @ F variable = value1, value2, … up to 10 values**

The F variable is a discrete variable that may take up to 10 values. In the first form, the variable is inspected and the sample is partitioned according to the values found. In the second case, specific values are used – this case might exclude some of the sample. For example, suppose *educ* were coded 12, 16, 20 and you specified **; Effects: income @ educ = 26,20**. Then, the analysis would be done for the two parts of the sample with *educ* = 16 and 20 while observations with *educ* = 12 would not be used in the analysis.

This specification operates differently from the | specification. Using | Z variable, you manipulate the values in the sample. Using @ F variable, you select observations, but do not change the actual values used in the data. All three specifications may be combined to produce counterfactuals. For example, the following specification

; Effects: income @ sex = 1 | sex = 0 & age = 20(5)80

selects the part of the sample that is female and computes the partial effects for that subsample while assuming that they are male. That is, the @ sex = 1 specification specifies the subsample for which *sex* = 1 (*female*) and computes the partial effect of income for ages from 20 to 80 by 5, while setting the sex dummy variable to 0 (*male*). The results produced are

```
======================================================================
Subsample for this iteration is SEX      =  1   Observations:    2170
======================================================================
----------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
----------------------------------------------------------------------
Effects on function with respect to INCOME
Results are computed by average over sample observations
Partial effects for continuous INCOME   computed by differentiation
Effect is computed as derivative     = df(.)/dx
----------------------------------------------------------------------
df/dINCOME         Partial    Standard
(Delta method)     Effect      Error     |t|  95% Confidence Interval
----------------------------------------------------------------------
Partial effect     -.05791     .05847     .99     -.17251       .05669
----------------------------------------------------------------------
SEX     =   .00 ------------------------------------------------------
AGE     = 20.00    -.01761     .00817    2.15     -.03362      -.00159
AGE     = 25.00    -.02343     .01070    2.19     -.04439      -.00246
AGE     = 30.00    -.02901     .01321    2.20     -.05489      -.00312
AGE     = 35.00    -.03341     .01529    2.19     -.06338      -.00345
AGE     = 40.00    -.03606     .01664    2.17     -.06867      -.00345
AGE     = 45.00    -.03687     .01717    2.15     -.07052      -.00322
AGE     = 50.00    -.03608     .01696    2.13     -.06932      -.00284
AGE     = 55.00    -.03405     .01616    2.11     -.06572      -.00238
AGE     = 60.00    -.03109     .01490    2.09     -.06030      -.00188
AGE     = 65.00    -.02742     .01330    2.06     -.05349      -.00134
AGE     = 70.00    -.02320     .01145    2.03     -.04565      -.00076
AGE     = 75.00    -.01869     .00946    1.98     -.03723      -.00015
AGE     = 80.00    -.01424     .00747    1.91     -.02887       .00040
```

R11.4.8 Fixing Variables for the Entire Analysis

You might want to fix certain extraneous variables during the analysis, apart from the analysis. Use the specification(s)

; Fix = name [value], name [value], …

For example, if the model were expanded to

```
LOGIT    ; Lhs = doctor
         ; Rhs = one,age,age^2,income,sex, sex*income,hsat $
```

Where *hsat* is health satisfaction, and we then wish to analyze the partial effects in the model assuming for the present that everyone in the sample reported 10 (the highest value) for *hsat*, we could use something like

PARTIAL EFFECTS ; Fix = hsat [10] ; Effects: income & educ = 20(5)80 $

We note, there is potential for conflicts among these specifications. For example,

; Fix = hsat [10] ; Effects: income | hsat = 8,9,10

has an inconsistency. This will produce a diagnostic,

```
Conflict between ; Fix=... and scenario
```

However, it is not possible to catch all possible conflicts. It is necessary to be cautious when using the global setting to fix some variables.

R11.4.9 Saving Individual Partial Effects

You can save the individual specific partial effects computed when they are averaged by adding

; Save

to the command. If you have a compound scenario, the value that is saved is the first one computed. It is best to use this only with a simple partial effects computation. The operation creates a new variable named *partlfct*. The estimated standard error is also saved, as *se_partl*. For example,

```
-->PARTIAL EFFECTS ; Effects: income  ; save $
----------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
----------------------------------------------------------------------
Effects on function with respect to INCOME
Results are computed by average over sample observations
Partial effects for continuous INCOME   computed by differentiation
Effect is computed as derivative     = df(.)/dx
Effects are saved as variable PARTLFCT. Std.Errors as SE_PARTL
----------------------------------------------------------------------
df/dINCOME         Partial    Standard
(Delta method)     Effect      Error     |t|  95% Confidence Interval
----------------------------------------------------------------------
APE. Function      -.04811     .02206    2.18     -.09135     -.00487
```

If you now use **DSTAT ; Rhs = partlfct $** the sample mean reported will be -0.08905. However, the sample standard deviation will not equal -0.04811. The value above is the standard deviation computed using the delta method, not the sample standard deviation of the computed values. The standard errors computed for each observation using the delta method are also saved in the variable *se_partl*.

R11.4.10 Partial Effects for Categorical Variables

The set of results discussed here are based on Hodge and Shankar (2014). Categorical variables appear in many applications. For an example, consider a model in which education appears not as years, but as levels, lths, hs, college and graduate. The application will typically be based either on a ‘category variable,’ say level coded 0 to 3 (or 1 to 4), or a set of dummy variables for the four categories. In either case, in actually computing the model, three of the four dummy variables will appear in the specification – the ‘base case’ will typically be either the highest or the lowest category. Also typically, partial effects in the model will be computed for each included dummy variable, relative to the base case. The conceptual experiment involved in that computation would be a change, or transition, from the base case to the included category. In the example given, for example, the partial effect for the ‘college’ category would be the effect on the expected outcome of transition from the less than high school category to the college category, skipping the high school category. One can easily imagine other useful experiments, such as the partial effect of college after high school. This partial effect is easily computed by subtracting the high school partial effect from the college partial effect. *LIMDEP* allows you to automate this calculation by computing ‘transition matrices’ for categorical variable.

The procedure for this calculation operates in the following steps.

Step 1. A namelist is defined that contains all of the category dummy variables.

Step 2. The model is fit using the namelist in Step 1 on the Rhs. To avoid the dummy variable trap, one of the categories is dropped. You do this in your model command by following the namelist name by a dot on the Rhs list.

Step 3. Use **PARTIALS ; Effects: the namelist ; Transition $** (Note that the full namelist is used here.)

In the following example, the indicated four levels of education are used in the logit model for visiting the doctor at least once.

```
RECODE      ; educ -> edlevel ; 7/11 = 0; 11/12 = 1; 12/16 = 2; * = 3 $
CREATE      ; lths = (edlevel = 0) ; hs = (edlevel = 1)
            ; college = (edlevel = 2) ; gradschl = (edlevel = 3) $
NAMELIST    ; edlevels = lths,hs,college,gradschl $
LOGIT       ; Lhs = doctor ; Rhs = one,income,edlevels.$
PARTIALS    ; Effects: edlevels ; Transition $
```

The following results are produced:

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit:Probability(DOCTOR=1)
---------------------------------------------------------------------
Effects of switches between categories in EDLEVELS (dummy variables)
Results are computed by average over sample observations
LTHS    = .5600   HS      = .2131   COLLEGE = .1583   GRADSCHL= .0687
---------------------------------------------------------------------
df/dEDLEVELS       Partial    Standard
From --> To        Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
LTHS     LTHS       .00000     .00000     .00      .00000      .00000
LTHS     HS       -.01481     .00746    1.98     -.02943     -.00018
LTHS     COLLEGE   -.03486     .00846    4.12     -.05144     -.01828
LTHS     GRADSCHL  -.11770     .01250    9.42     -.14219     -.09321
LTHS     (Other)   -.03808     .00604    6.30     -.04992     -.02623
HS       LTHS       .01481     .00746    1.98      .00018      .02943
HS       HS         .00000     .00000     .00      .00000      .00000
HS       COLLEGE   -.02005     .00975    2.06     -.03916     -.00095
HS       GRADSCHL  -.10289     .01330    7.74     -.12895     -.07683
HS       (Other)   -.00247     .00713     .35     -.01646      .01151
COLLEGE  LTHS       .03486     .00846    4.12      .01828      .05144
COLLEGE  HS         .02005     .00975    2.06      .00095      .03916
COLLEGE  COLLEGE    .00000     .00000     .00      .00000      .00000
COLLEGE  GRADSCHL  -.08284     .01379    6.01     -.10987     -.05581
COLLEGE  (Other)    .02151     .00810    2.65      .00563      .03740
GRADSCHL LTHS       .11770     .01250    9.42      .09321      .14219
GRADSCHL HS         .10289     .01330    7.74      .07683      .12895
GRADSCHL COLLEGE    .08284     .01379    6.01      .05581      .10987
GRADSCHL GRADSCHL   .00000     .00000     .00      .00000      .00000
GRADSCHL (Other)    .10839     .01217    8.91      .08454      .13223
---------------------------------------------------------------------
* (Other) = conditional share weighted average of all switch effects
---------------------------------------------------------------------
```

Since there are four categories, there are 12 possible transitions (from each category to each different category). *LIMDEP* computes an additional transition effect based on the experiment that the individual leaves a given category, but it is not known where they go. In that case, the destination is computed as a sample shares weighted average of the other possible destinations. Finally, the transition matrix, itself, is produced as a summary of the effects in the main table.

```
-----------------------------------------------------------------
Partial Effects Transition Matrix for EDLEVELS
There are  4 categories (sample %)
  01=LTHS     (56.00) 02=HS       (21.31) 03=COLLEGE  (15.83)
  04=GRADSCHL ( 6.87)
Entry = effect on outcome of switch from row category to column
Switch to Other is unspecified switch out of row category
--------+--------------------------------------------------------
        |     01     02     03     04  Other
--------+--------------------------------------------------------
    LTHS|   .000  -.015  -.035  -.118  -.038
      HS|   .015   .000  -.020  -.103  -.002
 COLLEGE|   .035   .020   .000  -.083   .022
GRADSCHL|   .118   .103   .083   .000   .108
--------+--------------------------------------------------------
```

The category variable may appear directly in the model and the partial effects. For example, the preceding has a four level category variable, *edlevel*, coded 0,1,2,3. The **LOGIT** command with *#edlevel* in it directly produces

LOGIT ; Lhs = doctor ; Rhs = one,income,#edlevel $

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
Dependent variable                DOCTOR
Log likelihood function    -17960.85188
Restricted log likelihood  -18019.55173
Chi squared [  4](P= .000)    117.39970
Significance level               .00000
McFadden Pseudo R-squared      .0032576
Estimation based on N =  27326, K =   5
Inf.Cr.AIC  =  35931.7 AIC/N =    1.315
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .64734***      .02874    22.53  .0000      .59102    .70367
  INCOME|    -.13129*        .07293    -1.80  .0718     -.27424    .01165
--------+--------------------------------------------------------------------
 EDLEVEL| Base = 1
      2 |    -.06416**       .03222    -1.99  .0464     -.12731   -.00101
      3 |    -.14940***      .03591    -4.16  .0000     -.21978   -.07902
      4 |    -.48820***      .05063    -9.64  .0000     -.58744   -.38896
--------+--------------------------------------------------------------------
```

We follow the model command with

PARTIALS ; Effects: #edlevel ; transition
; Category = lths,hs,college,gradschl $

The partial effects are identical to those given earlier when the namelist was used to provide the dummy variables for the categories.

R11.4.11 Computing Partial Effects at Sample Means

The default form of effect is the 'average partial effect.' The effect is computed by computing the derivative function for each observation in the sample. The alternative approach is to compute the effect at the means of the data by adding

; Means

to the command. This changes the results to those below. Note, the second line of the legend in the example below indicates how the effects are computed.

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to AGE
Results are computed at sample means of all variables
Partial effects for continuous AGE      computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dAGE            Partial    Standard
(Delta method)     Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
Partial effect      .00535     .00070    7.66      .00398      .00672
```

When you use the '@' specification, the sample means are recomputed for the various subsamples. For example,

; Effects: educ @ female = 0,1 (male,female) | age = 20,30,40 ; Means

computes the partial effect and the scenario at the sample means for *males*, then for *females*.

```
---------------------------------------------------------------------
Partial Effects  Analysis for Logit Probability Function
---------------------------------------------------------------------
Effects on function with respect to EDUC
Results are computed at sample means of all variables
Partial effects for continuous EDUC     computed by differentiation
Effect is computed as derivative     = df(.)/dx
---------------------------------------------------------------------
df/dEDUC           Partial    Standard
(Delta method)     Effect      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
Subsample for this iteration is FEMALE   =  0   Observations:    2311
Partial effect     -.00084     .00322     .26     -.00715      .00546
---------------------------------------------------------------------
AGE     = 20.00 -----------------------------------------------------
Effect at means    -.00090     .00345     .26     -.00766      .00585
---------------------------------------------------------------------
AGE     = 30.00 -----------------------------------------------------
Effect at means    -.00088     .00336     .26     -.00746      .00570
---------------------------------------------------------------------
AGE     = 40.00 -----------------------------------------------------
Effect at means    -.00085     .00325     .26     -.00722      .00552
---------------------------------------------------------------------
Subsample for this iteration is FEMALE   =  1   Observations:    2170
Partial effect     -.00083     .00315     .26     -.00700      .00535
---------------------------------------------------------------------
AGE     = 20.00 -----------------------------------------------------
Effect at means    -.00090     .00342     .26     -.00761      .00581
---------------------------------------------------------------------
AGE     = 30.00 -----------------------------------------------------
Effect at means    -.00087     .00333     .26     -.00739      .00565
---------------------------------------------------------------------
AGE     = 40.00 -----------------------------------------------------
Effect at means    -.00084     .00321     .26     -.00713      .00544
```

R11.4.12 Weighted Observations

Sample means and averages of partial effects (and simulations) are obtained by simple averages of the sample observations. You may supply sample weights as usual with

; Wts = weighting variable

(See Section R8.6.) When the scenario specifies a partitioning of the sample using '@specification,' the weights are scaled to sum to the number of observations in the subsample.

R11.4.13 Robust Covariance Matrices

PARTIAL EFFECTS does not compute a covariance matrix. It uses the one provided by the last model estimated, or provided by you in your function definition. If your estimated model included a robust (e.g., cluster corrected) covariance matrix, then the standard errors and confidence intervals will be similarly robust.

R11.4.14 Changing the Model Analyzed by PARTIAL EFFECTS

The function that *LIMDEP* uses for **PARTIAL EFFECTS** is the model left behind by the previous model command. The model will remain in place until another fitted model changes its place. However, you can specify your own model, or function – it need not be a model; this can be any function that you wish to analyze. The additional information in the command is

```
; Function    = any user defined function
; Covariance  = matrix
; Parameters  = set of values
; Labels      = names of parameters
```

The function definition is any function that you wish to specify using the same form as **MAXIMIZE**, **NLSQ**, **WALD**, etc. The function is assumed to involve an estimated parameter vector for which you also have in hand an estimated covariance matrix. The labels are provided so that you can differentiate between parameters and all the other numeric entities that can appear in the function.

To consider a perhaps contrived example, suppose we had fit a probit model and were interested in examining the behavior of the hazard function. The model is

$$\text{Prob}(y = 1 \mid \mathbf{x}) = \Phi(\boldsymbol{\beta}'\mathbf{x}).$$

The hazard function is

$$h(\boldsymbol{\beta}'\mathbf{x}) = -\text{dln}\Phi(-\boldsymbol{\beta}'\mathbf{x})/\text{d}(\boldsymbol{\beta}'\mathbf{x}) = \phi(\boldsymbol{\beta}'\mathbf{x})/[1 - \Phi(\boldsymbol{\beta}'\mathbf{x})].$$

This is not a conditional mean function, but it might nonetheless be interesting. To continue our example, we will employ this template and examine the partial effect of income on the hazard function for a probit model. The income variable in our model enters the function nonlinearly in several terms. Step 1 is definition and estimation of the model.

```
NAMELIST    ; xprobit = one,age,educ,income,income^2,age*income,hsat $
PROBIT      ; Lhs = doctor ; Rhs = xprobit $
```

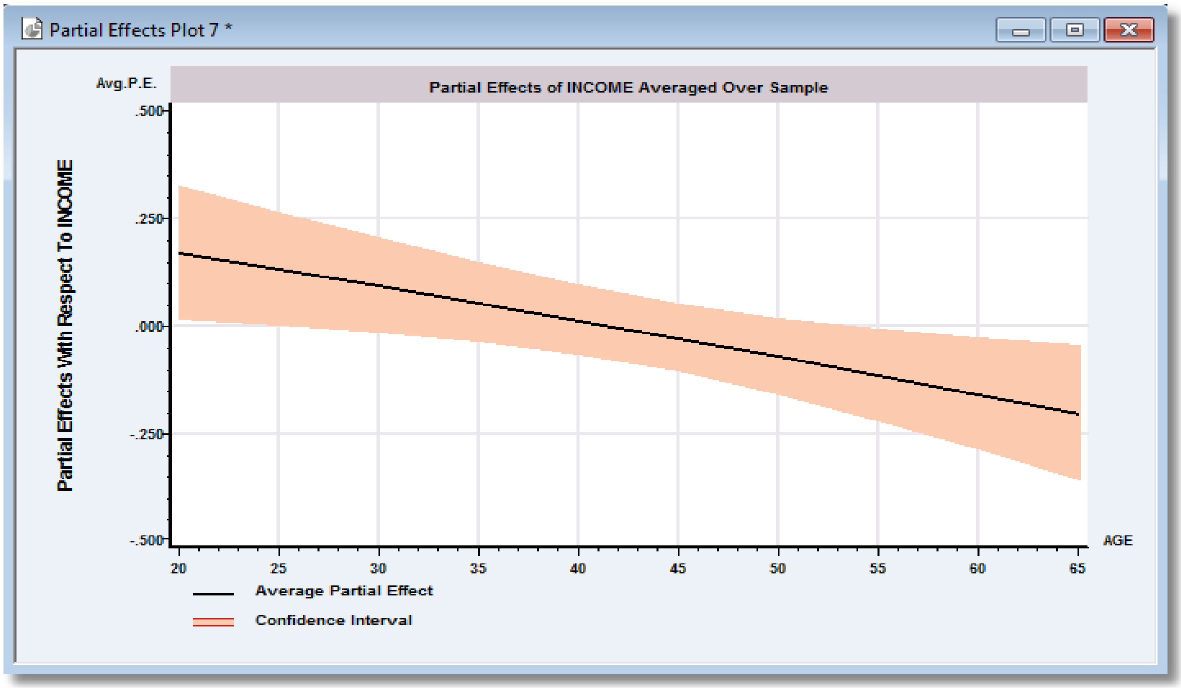
```
Normal exit:   4 iterations. Status=0, F=    2727.435
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               DOCTOR
Log likelihood function     -2727.43478
Restricted log likelihood   -2908.96085
Chi squared [   6 d.f.]       363.05212
Significance level               .00000
McFadden Pseudo R-squared      .0624024
Estimation based on N =   4481, K =   7
Inf.Cr.AIC  = 5468.870 AIC/N =    1.220
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|     .86475***      .23176     3.73  .0002      .41050   1.31899
     AGE|     .01642***      .00422     3.89  .0001      .00815    .02469
    EDUC|    -.00166         .00872     -.19  .8494    -.01875    .01544
  INCOME|     .83966         .59137     1.42  .1557    -.31941   1.99874
        |Constructed variable INCOME^2.0
_ntrct01|    -.06449         .25065     -.26  .7970    -.55576    .42678
        |Interaction AGE*INCOME
_ntrct02|    -.02484**       .01176    -2.11  .0346    -.04789   -.00179
    HSAT|    -.15719***      .00985   -15.95  .0000    -.17650   -.13787
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Step 2 is estimation of the partial effects.

```
PARTIAL EFFECTS  ; Labels = b1,b2,b3,b4,b5,b6,b7
                 ; Parameters = b
                 ; Covariance  = varb
                 ; Function =  bx  = b1'xprobit |
                               cdf = Phi(bx)     |
                               pdf = N01(bx)     |
                               pdf/(1-cdf)
                 ; Effects: income & age = 20(5)65
                 ; Plot(ci) $
```

The function is defined recursively purely for convenience. The same results would be produced by **; Function = N01(b1'x)/(1 - Phi(b1'x))**; we decomposed it above to illustrate how to compute a complicated function in parts. The estimated effects and a plot with 95% confidence limits are as follows:

```
----------------------------------------------------------------------
Partial Effects  Analysis for User Specified Function
----------------------------------------------------------------------
Effects on function with respect to INCOME
Results are computed by average over sample observations
Partial effects for continuous INCOME   computed by differentiation
Effect is computed as derivative     = df(.)/dx
----------------------------------------------------------------------
df/dINCOME          Partial    Standard
(Delta method)      Effect      Error      |t|  95% Confidence Interval
----------------------------------------------------------------------
APE. Function       -.02229     .03937      .57     -.09945       .05488
AGE     = 20.00      .16945     .07867     2.15      .01525       .32366
AGE     = 25.00      .13148     .06693     1.96      .00030       .26266
AGE     = 30.00      .09251     .05597     1.65     -.01719       .20220
AGE     = 35.00      .05257     .04670     1.13     -.03895       .14410
AGE     = 40.00      .01171     .04067      .29     -.06801       .09143
AGE     = 45.00     -.03005     .03980      .76     -.10805       .04795
AGE     = 50.00     -.07268     .04470     1.63     -.16028       .01493
AGE     = 55.00     -.11613     .05407     2.15     -.22211      -.01014
AGE     = 60.00     -.16038     .06625     2.42     -.29023      -.03052
AGE     = 65.00     -.20539     .08013     2.56     -.36244      -.04835
```



R11.4.15 Technical Details: The Delta Method and Krinsky and Robb

PARTIAL EFFECTS accounts for all interactions and nonlinearities built into the model or function specification. The feature is available for every model fit by the program. Based on the formulation in Section R11.4.11, you can use this process with any function that you can compute with the data in your sample, whether the function is a model or anything else. It is also independent of the sample used to fit the model. The parameter vector and associated covariance matrix are used to compute functions of your data. No connection is assumed between the estimation sample and functions you compute. You can, for example, fit a model with a given sample, then change to a different set of observations and analyze the partial effects with respect to that second sample.

The computations proceed as follows:

Step 1. Set subsamples: This is defined by @ values in Section R11.4.7

Step 2. Do for observations in the subsample:

Step 3. Obtain full observation $\mathbf{x}(i)$ from raw data set.

- Fix any values in $\mathbf{x}(i)$ as prescribed by Section R11.4.8.
- Fix any specific values by | or & specified by Section R11.4.5.
- Compute any interactions defined by the model.
- Perturb the original $\mathbf{x}(i)$ then recompute the interactions.
- Compute derivatives of functions with respect to $\mathbf{x}(i)$; partial effects.
- Perturb parameters and compute Jacobian for delta method.
- Accumulate average function, average derivatives, average Jacobian.

Step 4. Obtain appropriate asymptotic covariance matrix using covariance matrix and average Jacobian.

The structure of the iteration implies that the interaction terms are computed after the data are perturbed. Thus, if the model contains x and x^2, the derivatives are obtained by evaluating the function first at (x,x^2), then at $[(x+\Delta x), (x+\Delta x)^2]$, which produces the right result for the partial effect with respect to x. When the effects is with respect to a dummy variable, the perturbation step consists of fixing the variable at 1 then 0. When there are multiple evaluations of the effects, as in our earlier examples, the iteration takes place over Steps 2-4. The entire operation is carried out with the sequence of values specified with | or &.

Every parametric model fit by *LIMDEP* leaves behind a 'last model function.' This will usually be a prediction function such as a conditional mean. Familiar examples are the index function, $\boldsymbol{\beta}'\mathbf{x}$, for the linear regression, the probabilities for the probit model, $\Phi(\boldsymbol{\beta}'\mathbf{x})$, or the logit model, $\Lambda(\boldsymbol{\beta}'\mathbf{x})$, and the conditional mean for the Poisson and negative binomial models, $\exp(\boldsymbol{\beta}'\mathbf{x})$. There are many others, such as the compound conditional mean functions, for the zero inflated Poisson model or the sample selection model. We denote this function generally as $H(\mathbf{x},\boldsymbol{\beta})$ where $\mathbf{x}$ is the observation vector that includes all variables in the model (both dependent and independent and $\boldsymbol{\beta}$ is the full parameter vector.

The specific model functions that apply for each model are described in the *Econometric Modeling Guide*. Let $\hat{\boldsymbol{\beta}}$ be the sample estimate of **β**, and let $\hat{\boldsymbol{\Sigma}}$ be the estimate of the asymptotic covariance matrix of $\hat{\boldsymbol{\beta}}$. These, with the specification of the model, itself, constitute the last model function noted above. (Note in the process described in Section R11.4.11, you provide these explicitly with your command.) The average partial effects reported by **PARTIAL EFFECTS** are computed as

$$\hat{\delta}(x) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial H\left(\mathbf{x},\hat{\boldsymbol{\beta}}\right)}{\partial x} \text{ or}$$
$$\hat{\Delta}(x) = \frac{1}{N}\sum_{i=1}^{N} \left[H\left(\mathbf{x},\hat{\boldsymbol{\beta}} \mid x=1\right) - H\left(\mathbf{x},\hat{\boldsymbol{\beta}} \mid x=0\right)\right].$$

Derivatives for continuous variables are computed numerically. The Jacobian required to apply the delta method is

$$\hat{\mathbf{J}}(x) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial^2 H\left(\mathbf{x}_i,\hat{\boldsymbol{\beta}}\right)}{\partial x_i \partial \hat{\boldsymbol{\beta}}'} \text{ or}$$
$$\hat{\mathbf{J}}(d) = \frac{1}{N}\sum_{i=1}^{N} \left[\frac{\partial H\left(\mathbf{x}_i,\hat{\boldsymbol{\beta}} \mid d=1\right)}{\partial \hat{\boldsymbol{\beta}}'} - \frac{\partial H\left(\mathbf{x}_i,\hat{\boldsymbol{\beta}} \mid d=0\right)}{\partial \hat{\boldsymbol{\beta}}'}\right].$$

The elements of the Jacobian are computed numerically. The estimator of the asymptotic variance for a particular partial effect is then $\hat{\mathbf{J}}\,\hat{\boldsymbol{\Sigma}}\,\hat{\mathbf{J}}'$.

METHODOLOGICAL NOTE: The computation of the asymptotic variance of the partial effects used when the delta method is employed assumes that the exogenous data are given – they are treated as constants. That is, the analysis is done conditionally on the data. No attempt is made to correct the variance of the parameter estimator to account for the possibility that the variation in the current sample might be different from that in the estimation sample.

The method of Krinsky and Robb (1986) can be requested instead of the delta method for any partial effect (or simulation as described in Chapter R12). The method of Krinsky and Robb operates as follows:

Step 1. Estimate the parameters of the model and $\hat{\boldsymbol{\Sigma}}$ as discussed above.

Step 2. For R repetitions,

a. Generate a sampled value of $\hat{\boldsymbol{\beta}}$ (r) by drawing a vector from the normal distribution with mean vector $\hat{\boldsymbol{\beta}}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}$.

b. Compute the partial effect, $\hat{\delta}(x)$ [r] using $\hat{\boldsymbol{\beta}}$ (r)

c. Compute the empirical standard deviation of the R replicates of $\hat{\delta}(x)$ [r].

The result in Step 2a is the alternative to the asymptotic standard based on the delta method. There is no theoretical argument that the delta method is better or worse than Krinsky and Robb. They are two methods of computing an estimated standard error based on the asymptotic normal distribution of the estimator of the parameters. The delta method relies on a linear approximation to the partial effect. The Krinsky and Robb method does not. To request Krinsky and Robb, you add **; K&R** to the command and, if desired, the value of R, as follows:

; K&R ; Pts = the desired value of R.

The default value of R is 250. You may supply any alternative. Note that larger R implies longer computation time.

R11.5 Partial Effects Estimated with Models

The model specification is generally of the form

Model ; Lhs = … ; Rhs = list of variables, x $

In most of the index function models such as regressions, probit, tobit, logit and Poisson models, the partial effects are of the form

$$\boldsymbol{\delta}(\boldsymbol{\beta}'\mathbf{x},\boldsymbol{\theta}) = \partial E[y|\boldsymbol{\beta}'\mathbf{x},\boldsymbol{\theta}]/\partial\mathbf{x} = g_1(\boldsymbol{\beta}'\mathbf{x},\boldsymbol{\theta}) \times \boldsymbol{\beta},$$

where $g_1(\boldsymbol{\beta}'\mathbf{x},\boldsymbol{\theta})$ is a scale factor that involves the data and all the model parameters. The unattached parameter, $\boldsymbol{\theta}$, might be the standard deviation, σ in a tobit model or a correlation coefficient, ρ, in a sample selection model. These are usually computed by averaging terms across the sample observations, but in some cases, at the means of the full sample instead. The model may also contain additional lists of variables. For example in a model of heteroscedasticity, an additional function might appear, such as

; Hfn = list of variables z.

The conditional mean or other function that will be analyzed will then be of the form

$$g(\mathbf{x},\mathbf{z}) = g(\boldsymbol{\beta}'\mathbf{x}, \boldsymbol{\gamma}'\mathbf{z}, \boldsymbol{\theta})$$

where $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$ and $\boldsymbol{\theta}$ are sets of coefficients. Partial effects will take the form of scaled coefficients

$$\partial g(\boldsymbol{\beta}'\mathbf{x}, \boldsymbol{\gamma}'\mathbf{z}, \boldsymbol{\theta})/\partial\mathbf{x} = g_1(\boldsymbol{\beta}'\mathbf{x}, \boldsymbol{\gamma}'\mathbf{z}, \boldsymbol{\theta})\boldsymbol{\beta},$$
$$\partial g(\boldsymbol{\beta}'\mathbf{x}, \boldsymbol{\gamma}'\mathbf{z}, \boldsymbol{\theta})/\partial\mathbf{z} = g_2(\boldsymbol{\beta}'\mathbf{x}, \boldsymbol{\gamma}'\mathbf{z}, \boldsymbol{\theta})\gamma.$$

If a variable, *w*, appears in both **x** and **z**, then the effects are added to get the partial effect of *w*. In some cases, the two parts might be of interest. For example, in a recursive bivariate probit model, we can identify separate 'direct' and 'indirect' effects for some variables.

Partial effects for nearly all of the regression, discrete choice, and limited dependent variable models (including, for example, multinomial logit models) have been hard coded into *LIMDEP*. A full set of output (estimate, standard error, t-ratio, prob value, confidence interval) is then reported. Estimates are computed either by averaging observations or at the overall means of the data set and optionally for the group means for a discrete variable that you may provide (with up to 10 levels). To obtain the partial effects based on the observations used to fit the model, use

; Partial Effects

In most cases, the partial effects are computed by averaging the effects across observations, producing 'average partial effects.' You can obtain the calculation done specifically at the sample means of the data by adding

; Means

to the command.

R11.5.1 Partial Effects for Single Index Models

The first case noted above will apply in most applications. The following illustrates for a Poisson regression model

```
POISSON     ; Lhs = docvis ; Rhs = one,age,educ,married
            ; Partial Effects $
```

```
-----------------------------------------------------------------------------
Poisson Regression
Dependent variable                DOCVIS
Log likelihood function   -105028.97738
Restricted log likelihood -108662.13583
Chi squared [   3](P= .000)   7266.31690
Significance level                .00000
McFadden Pseudo R-squared       .0334354
Estimation based on N =   27326, K =   4
Inf.Cr.AIC  = 210066.0 AIC/N =     7.687
Chi- squared =259133.59600  RsqP= .0674
G  - squared =157019.37395  RsqD= .0442
Overdispersion tests: g=mu(i)  : 22.067
Overdispersion tests: g=mu(i)^2: 22.218
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
  DOCVIS|  Coefficient      Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .86184***     .02582    33.37  .0000      .81123    .91246
     AGE|     .02188***     .00031    70.97  .0000      .02127    .02248
    EDUC|    -.05426***     .00167   -32.53  .0000     -.05753   -.05099
 MARRIED|    -.11071***     .00802   -13.80  .0000     -.12644   -.09499
--------+--------------------------------------------------------------------
***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
Partial derivatives of expected val. with
respect to the vector of characteristics.
Effects are averaged over individuals.
Observations used for means are All Obs.
Conditional Mean at Sample Point   2.8728
Scale Factor for Marginal Effects  2.8728
--------+--------------------------------------------------------------------
        |     Partial      Standard            Prob.      95% Confidence
  DOCVIS|      Effect        Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     AGE|     .06965***      .00101    69.00  .0000       .06767    .07162
    EDUC|    -.17274***      .00534   -32.33  .0000     -.18321   -.16227
 MARRIED|    -.36333***      .02714   -13.39  .0000     -.41652   -.31013   #
--------+--------------------------------------------------------------------
#  Partial effect for dummy variable is E[y|x,d=1] - E[y|x,d=0]
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

There are cases in which there is no appropriate conditional mean function to differentiate. The ordered choice models are a leading case. In general, for models of probabilities, *LIMDEP* computes partial effects of the implied probabilities. In the example below, the dependent variable in the ordered probit model takes values 0,1,2, so there are three sets of partial effects.

OPROBIT ; Lhs = hlthsat ; Rhs = one,age,educ,married
; Partial Effects $

```
-----------------------------------------------------------------------------
Ordered Probability Model
Dependent variable              HLTHSAT
Log likelihood function     -3170.58904
Restricted log likelihood   -3291.50941
Chi squared [   3 d.f.]       241.84074
Significance level               .00000
McFadden Pseudo R-squared      .0367371
Estimation based on N =   4481, K =   5
Inf.Cr.AIC  = 6351.178 AIC/N =    1.417
Underlying probabilities based on Normal
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
 HLTHSAT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|    2.54306***      .14140    17.98  .0000     2.26592   2.82019
     AGE|    -.02164***      .00171   -12.64  .0000     -.02499   -.01828
    EDUC|     .05600***      .00834     6.71  .0000      .03965    .07234
 MARRIED|     .03001         .04450      .67  .5000     -.05720    .11723
        |Threshold parameters for index
   Mu(1)|    1.99085***      .05104    39.01  .0000     1.89082   2.09088
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Marginal effects for ordered probability model
M.E.s for dummy variables are Pr[y|x=1]-Pr[y|x=0]
Names for dummy variables are marked by *.
-----------------------------------------------------------------------------
```

```
--------+-------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
 HLTHSAT|      Effect    Elasticity      z     |z|>Z*         Interval
--------+-------------------------------------------------------------------
        |--------------[Partial effects on Prob[Y=00] at means]--------------
     AGE|     .00066***      2.40368     7.68  .0000       .00049    .00083
    EDUC|    -.00172***     -1.63481    -5.51  .0000      -.00233   -.00111
*MARRIED|    -.00094         -.07809     -.66  .5081      -.00371    .00184
        |--------------[Partial effects on Prob[Y=01] at means]--------------
     AGE|     .00765***       .86788    12.72  .0000       .00647    .00883
    EDUC|    -.01980***      -.59027    -6.72  .0000      -.02557   -.01403
*MARRIED|    -.01062         -.02773     -.67  .5002      -.04149    .02025
        |--------------[Partial effects on Prob[Y=02] at means]--------------
     AGE|    -.00831***      -.59691   -12.72  .0000      -.00959   -.00703
    EDUC|     .02152***       .40597     6.72  .0000       .01524    .02779
*MARRIED|     .01156          .01910      .67  .5008      -.02209    .04520
--------+-------------------------------------------------------------------
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
----------------------------------------------------------------------------
```

When partial effects are reported for probability models such as the ordered probit model above, *LIMDEP* also reports the elasticities of the probabilities with respect to the independent variables. These are

$$\text{Elasticity} = \partial \ln\text{Prob}(\ldots)/\partial \ln X \;=\; [\text{Mean } X/ \text{ Mean Prob}(\ldots)] \times \text{Partial effect.}$$

The footnote indicates that the confidence interval is given for the partial effect, not the elasticity.

R11.5.2 Partial Effects for Dummy Rhs Variables

Models will often involve binary variables. The marginal effects described in the preceding sections are computed by differentiating the expected value function with respect to the variables in the model. But, one cannot actually differentiate with respect to a dummy variable, and an appropriate way to compute an effect for a dummy variable is to compare the values of the function with the binary variable set to one and zero. The appropriate effect for dummy variable z is, then

$$\begin{aligned}\delta_{10} &= \text{Effect of dummy variable } z\\ &= f(\ldots|\text{other variables}, z = 1) \;-\; f(\ldots|\text{other variables}, z = 0)\\ &= f_1(\boldsymbol{\beta}|\mathbf{x}, z = 1) - f_0(\boldsymbol{\beta}|\mathbf{x}, z = 0).\end{aligned}$$

In order to obtain the appropriate standard error for this estimate, one would then use the delta method applied to this function of the parameters, rather than the one shown earlier. Thus, the asymptotic variance for this estimator would be

$$\text{Asy.Var}[d_{01}] \;=\; \mathbf{g}_{10}\mathbf{S}\mathbf{g}_{10}'$$

where

$$\mathbf{g}_{10} = [\partial f_1(\boldsymbol{\beta}|\mathbf{x},z = 1)/\partial\boldsymbol{\beta}'] \;-\; [\partial f_0(\boldsymbol{\beta}|\mathbf{x},z = 1)/\partial\boldsymbol{\beta}'] \text{ (note, a row vector),}$$

and **S** is as defined earlier. This computation is automated in a few cases. The probit, ordered probit and Poisson models shown above all contain dummy variables that are autodetected by the partial effects program. The **PARTIAL EFFECTS** command described in Section R11.4 detects this automatically in all cases.

R11.5.3 Standard Errors and Confidence Intervals

Covariance matrices for marginal effects are computed using the delta method. When the estimated effects vector (using the estimated parameters) is **d**(**x**,**b**,**q**), we use

$$\text{Est.Var}[\mathbf{d}(\mathbf{x},\mathbf{b},\mathbf{q})] = \{\partial\, \mathbf{d}(\mathbf{x},\mathbf{q},\mathbf{b}) / \partial[\mathbf{b}'\ \mathbf{q}']\} \times \text{Est.Asy.Var}[\mathbf{b},\mathbf{q}] \times \{\partial\, \mathbf{d}(\mathbf{x},\mathbf{q},\mathbf{b}) / \partial[\mathbf{b}'\ \mathbf{q}']'\}.$$

For example, for the Poisson regression model (which has no $\boldsymbol{\theta}$),

$$\partial\boldsymbol{\delta}(\mathbf{x},\boldsymbol{\beta},\boldsymbol{\theta}) / \partial\boldsymbol{\beta}' = \partial\exp(\boldsymbol{\beta}'\mathbf{x})\boldsymbol{\beta} / \partial\boldsymbol{\beta}' = \exp(\boldsymbol{\beta}'\mathbf{x})\,[\,\mathbf{I} + \boldsymbol{\beta}\mathbf{x}'] = \boldsymbol{\Gamma},$$

which is estimated with **G** by computing $\boldsymbol{\Gamma}$ at **b** and the means of the regressors, while the estimated asymptotic covariance matrix is

$$\text{Est.Asy.Var}[\mathbf{b}] = (\mathbf{X}'\boldsymbol{\Lambda}\mathbf{X})^{-1} = \mathbf{S},\ \boldsymbol{\Lambda} = \text{diag}[\,\exp(\mathbf{b}'\mathbf{x})\,].$$

The standard errors for the reported marginal effects are then the square roots of the diagonal elements of **V** = **GSG′**. When the partial effects are computed by averaging over the sample observations, the preceding is modified by using the sample average, $\bar{\mathbf{G}} = (1/N)\Sigma_{i=1}^{N}\partial\mathbf{d}(\mathbf{b}'\mathbf{x}_i,\boldsymbol{\theta})/\partial\boldsymbol{\beta}'$.

R11.5.4 Significance Tests for Partial Effects

Marginal effects are reported with standard errors and ‘significance tests’ of their difference from zero in the same fashion as the coefficients. Whether one *should* test for significance in this fashion represents a gap in the orthodoxy. The raw coefficient in a model such as the Poisson regression does not represent the ‘effect’ of the respective ‘*x*’ on ‘*y*.’ Arguably, the marginal effect measures that. However, the marginal effect is a hodgepodge of all the coefficients (and some data) in the model. Testing for the significance of the effect is a qualitatively different exercise from testing the significance of a coefficient. Testing whether a coefficient is zero is equivalent to testing whether a variable is influential in the model, but testing whether a marginal effect is zero is not equivalent to that same test. Consider the results for a Poisson model based on the ship accident data examined in Greene (2012) given below. Note that based on the coefficient estimates, variable *T6064* is a ‘significant determinant’ of *acc*. But, the marginal effect of *T6064* on $E[y|\mathbf{x}]$ is not ‘significant,’ by the usual standard. Does this imply that *T6064* should be dropped from the model? Researchers differ on this question, but we think not.

```
-----------------------------------------------------------------------------
Poisson Regression
Dependent variable                     ACC
Log likelihood function          -72.82081
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
     ACC|  Coefficient      Error        z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|   -4.39059***      .71812    -6.11  .0000    -5.79808  -2.98310
 TYPE=01|    -.23881         .24102     -.99  .3218     -.71119    .23358
 TYPE=02|    -.54185*        .32155    -1.69  .0920    -1.17208    .08839
 TYPE=03|   -1.02222***      .34039    -3.00  .0027    -1.68937   -.35508
 TYPE=04|    -.38893         .30562    -1.27  .2032     -.98794    .21008
   T6064|    -.57984**       .23952    -2.42  .0155    -1.04930   -.11038
   T6569|     .12025         .20670      .58  .5607     -.28487    .52537
   T7074|     .28651         .19745     1.45  .1468     -.10048    .67349
  LOGMTH|     .87296***      .09947     8.78  .0000      .67800   1.06792
-----------------------------------------------------------------------------
Partial derivatives of expected val. With respect to the vector of
characteristics. Effects are averaged over individuals.
--------+--------------------------------------------------------------------
        |     Partial     Standard            Prob.      95% Confidence
     ACC|      Effect       Error        z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
 TYPE=01|   -2.29509       3.65390     -.63  .5299    -9.45660   4.86641   #
 TYPE=02|   -6.61880      10.05821     -.66  .5105   -26.33253  13.09492   #
 TYPE=03|   -7.10538***    2.74487    -2.59  .0096   -12.48522  -1.72554   #
 TYPE=04|   -3.45041       3.94534     -.87  .3818   -11.18313   4.28230   #
   T6064|   -5.31897       4.25823    -1.25  .2116   -13.66495   3.02701   #
   T6569|    1.28127       3.88184      .33  .7413    -6.32700   8.88953   #
   T7074|    3.18697       4.55099      .70  .4838    -5.73280  12.10674   #
  LOGMTH|    9.14039***    2.81756     3.24  .0012     3.61808  14.66271
--------+--------------------------------------------------------------------
#  Partial effect for dummy variable is E[y|x,d=1] - E[y|x,d=0]
```

The marginal effect is a function of all the coefficients in the model. The large standard error is a product of the mixture of all the coefficients and the highly nonlinear function, exp(.), which produces the relatively large matrix **G** in **V** = **GSG′**. In most, but far from all cases the ‘statistical significance’ of the marginal effects will be roughly the same as that of the corresponding coefficient. It turns out that the significance of the marginal effects is not a function of the point at which they are computed (whether the mean of the *x*s or some other point). (We draw on a useful study by Anderson and Newell (2003).) If the single index model is fit with the data measured in deviations from their means, the identical coefficients, but a different constant term, and the same asymptotic covariance matrix save for the row and column corresponding to the constant term will be produced. The marginal effects will be of the form $\mathbf{d} = f(a)\mathbf{b}$ where a is the estimated constant term (and is an element of **b**) because the part of the index function that corresponds to $\boldsymbol{\beta}'\mathbf{x}$ above will be zero for all terms save the constant. If we now expand the expression for the estimator asymptotic variance of d_k, we will find this to be

$$\text{Est.Asy.Var}[d_k] = [f(a)]^2\text{Est.Asy.Var}[b_k] + (f'(a)b_k)^2\text{Est.Asy.Var}[a] + 2f(a)b_k\text{Est.Cov}[a,b_k].$$

The ratio of d_k to its estimated standard error would be the same as that for b_k were it not for the second and third terms. So, in part, the statistical significance of the marginal effect for x_k hangs on the significance of the constant term, which seems hardly relevant to the question. While this clarifies the computation, we see this as a negative result.

So, should one report significance tests with (and for) marginal effects? This must be up to the researcher – we cannot answer the question here. In our opinion, based on the preceding, no. But, researchers still differ on this question. As such, *LIMDEP* reports standard errors and significance values for marginal effects. Whether they should be reported is up to the user.

R11.5.5 Partial Effects in Compound Models

Models are sometimes constructed in which variables enter in more than one place. Consider, for example, a probit model with exponential heteroscedasticity. The conditional mean in this model is

$$E[y|\mathbf{x},\mathbf{z}] = \text{Prob}[y=1|\mathbf{x},\mathbf{z}] = \Phi\,[\,\boldsymbol{\beta}'\mathbf{x} / \exp(\boldsymbol{\gamma}'\mathbf{z})\,]$$

where Φ denotes the standard normal CDF. In this model, the vectors of partial effects are

$$\partial E[y|\mathbf{x},\mathbf{z}] / \partial\mathbf{x} = \phi[\,\boldsymbol{\beta}'\mathbf{x} / \exp(\boldsymbol{\gamma}'\mathbf{z})\,]\,[1/\exp(\boldsymbol{\gamma}'\mathbf{z})] \times\boldsymbol{\beta},$$

$$\partial E[y|\mathbf{x},\mathbf{z}] / \partial\mathbf{z} = -\phi[\,\boldsymbol{\beta}'\mathbf{x} / \exp(\boldsymbol{\gamma}'\mathbf{z})\,]\,[\boldsymbol{\beta}'\mathbf{x} / \exp(\boldsymbol{\gamma}'\mathbf{z})] \times\boldsymbol{\gamma},$$

where ϕ denotes the standard normal density. These can be computed and tabulated separately. However, if **x** and **z** have any variables in common, then the marginal effect of that variable on the conditional mean is the sum of the two terms. Where this situation arises, *LIMDEP* computes the sum, then reports that value in both places, since the table of marginal effects is identified generally by variable. The following example shows the results for a heteroscedastic probit model.

```
PROBIT    ; Lhs = doctor
          ; Rhs = one,age,educ,married
          ; Heteroscedasticity
          ; Hfn = age,hhninc
          ; Partial Effects $
```

```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable                DOCTOR
Log likelihood function     -17647.13260
Restricted log likelihood   -18019.55173
Chi squared [  5](P= .000)     744.83827
Significance level                .00000
McFadden Pseudo R-squared       .0206675
Estimation based on N =  27326, K =   6
Inf.Cr.AIC  =  35306.3 AIC/N =    1.292
-----------------------------------------------------------------------------
```

```
--------+-------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+-------------------------------------------------------------------
        |Index function for probability.....................................
Constant|     .06445***      .01273     5.06  .0000       .03950     .08939
     AGE|     .00043         .00041     1.05  .2915     -.00037     .00124
    EDUC|    -.00363***      .00134    -2.71  .0067     -.00626    -.00101
 MARRIED|     .00407         .00286     1.42  .1553     -.00154     .00968
        |Variance function..................................................
     AGE|    -.04408***      .00621    -7.10  .0000     -.05625    -.03191
  INCOME|     .32827**       .14729     2.23  .0258      .03957     .61696
--------+-------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with
respect to the vector of characteristics
They are computed at the means of the Xs
Observations used for means are All Obs.
--------+-------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
  DOCTOR|      Effect    Elasticity      z    |z|>Z*         Interval
--------+-------------------------------------------------------------------
     AGE|     .00661**       .00279     2.37  .0177       .00115     .01207
    EDUC|    -.00847***      .00296    -2.86  .0042     -.01427    -.00267
 MARRIED|     .00949         .00660     1.44  .1505     -.00345     .02242
        |Variance function..................................................
     AGE|     .00661**       .00279     2.37  .0177       .00115     .01207
  INCOME|    -.00012         .01669     -.01  .9945     -.03283     .03260
--------+-------------------------------------------------------------------
Elasticity for a binary variable is marginal effect/Mean.
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
----------------------------------------------------------------------------
```

R11.5.6 Partial Effects in a Two Equation Model

In a few cases involving more than one dependent variable, *LIMDEP* creates an arrangement that is specific to the model at hand. The bivariate probit model is a common application in which each of the two equations may or may not have the multiplicative heteroscedasticity described above. In this case, there is no obvious conditional mean, and therefore, no obvious marginal effect. What is reported for this model is a conditional expectation, E[$y1|y2=1$] which is then a function of both $\boldsymbol{\beta}$ vectors and both $\boldsymbol{\gamma}$ vectors. The analysis shown in the following example decomposes the total partial effect.

The same addition is then done to get the total effect. This is shown in the table below, which then obtains the total effect:

```
BIVARIATE PROBIT  ; Lhs  = doctor,public
                  ; Rh1 = one,age,educ,married
                  ; Rh2 = one,educ,income,hhkids
                  ; Heteroscedasticity ; Hf1 = age,income
                  ; Partial Effects $
```

```
-----------------------------------------------------------------------------
FIML Estimates of Bivariate Probit Model
Dependent variable                DOCPUB
Log likelihood function    -25956.12488
Estimation based on N =  27326, K =  11
Inf.Cr.AIC  =  51934.2 AIC/N =    1.901
Disturbance model is multiplicative het.
Var. Parms follow   8 slope estimates.
For e(1),  2 estimates follow HHKIDS
--------+--------------------------------------------------------------------
  DOCTOR|                  Standard            Prob.      95% Confidence
  PUBLIC|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index    equation for DOCTOR........................................
Constant|     .06555***      .01432     4.58  .0000      .03749    .09361
     AGE|     .00047         .00079      .59  .5537     -.00108    .00202
    EDUC|    -.00374*        .00214    -1.74  .0810     -.00794    .00046
 MARRIED|     .00405         .00349     1.16  .2465     -.00280    .01089
        |Index    equation for PUBLIC........................................
Constant|    3.66184***      .05168    70.85  .0000     3.56054   3.76313
    EDUC|    -.17154***      .00405   -42.32  .0000     -.17949   -.16360
  INCOME|    -.98677***      .05061   -19.50  .0000    -1.08596   -.88758
  HHKIDS|    -.08149***      .02172    -3.75  .0002     -.12405   -.03892
        |Variance equation for DOCTOR........................................
     AGE|    -.04338***      .01086    -3.99  .0001     -.06467   -.02210
  INCOME|     .32184**       .14463     2.23  .0261      .03837    .60531
        |Disturbance correlation.............................................
RHO(1,2)|     .06707***      .01372     4.89  .0000      .04019    .09396
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

+--------------------------------------------------------+
|              Partial Effects for Ey1|y2=1              |
+----------+----------------------+----------------------+
|          | Regression Function  |  Heteroscedasticity  |
|          +----------------------+----------------------+
|          |   Direct | Indirect  |   Direct | Indirect  |
| Variable | Efct  x1 | Efct  x2  | Efct  h1 | Efct  h2  |
+----------+----------+-----------+----------+-----------+
|      AGE |   .00106 |   .00000  |   .00458 |   .00000  |
|     EDUC |  -.00845 |   .00121  |   .00000 |   .00000  |
|  MARRIED |   .00915 |   .00000  |   .00000 |   .00000  |
|   INCOME |   .00000 |   .00695  |  -.03397 |   .00000  |
|   HHKIDS |   .00000 |   .00057  |   .00000 |   .00000  |
+----------+----------+-----------+----------+-----------+

-----------------------------------------------------------------------------
Partial derivatives of  E[y1|y2=1]  with
respect to the vector of characteristics.
They are computed at the means of the Xs.
Effect shown is total of all parts above.
Estimate of E[y1|y2=1] = .613215
Observations used for means are  All Obs.
Total effects reported = direct+indirect.
-----------------------------------------------------------------------------
```

```
--------+-------------------------------------------------------------------
  DOCTOR|     Partial      Standard            Prob.      95% Confidence
  PUBLIC|      Effect        Error       z    |z|>Z*         Interval
--------+-------------------------------------------------------------------
     AGE|     .00564***      .00026    21.64  .0000      .00513    .00615
    EDUC|    -.00724***      .00138    -5.24  .0000     -.00995   -.00453
 MARRIED|     .00915         .00620     1.47  .1403     -.00301    .02131
  INCOME|    -.02702*        .01536    -1.76  .0786     -.05713    .00309
  HHKIDS|     .00057***      .00019     3.02  .0026      .00020    .00095
--------+-------------------------------------------------------------------
Partial derivatives of  E[y1|y2=1]  with
respect to the vector of characteristics.
Estimate of E[y1|y2=1] = .632895
Observations used for means are  All Obs.
These  are the  direct marginal  effects.
--------+-------------------------------------------------------------------
  DOCTOR|     Partial      Standard            Prob.      95% Confidence
  PUBLIC|      Effect        Error       z    |z|>Z*         Interval
--------+-------------------------------------------------------------------
     AGE|     .00564***      .00026    21.64  .0000      .00513    .00615
    EDUC|    -.00845***      .00136    -6.22  .0000     -.01111   -.00578
 MARRIED|     .00915         .00620     1.47  .1403     -.00301    .02131
  INCOME|    -.03397**       .01529    -2.22  .0263     -.06393   -.00400
  HHKIDS|        0.0    .....(Fixed Parameter).....
--------+-------------------------------------------------------------------
Partial derivatives of  E[y1|y2=1]  with
respect to the vector of characteristics.
Estimate of E[y1|y2=1] = .632895
Observations used for means are  All Obs.
These are the indirect  marginal effects.
--------+-------------------------------------------------------------------
  DOCTOR|     Partial      Standard            Prob.      95% Confidence
E[y1|x,z|      Effect        Error       z    |z|>Z*         Interval
--------+-------------------------------------------------------------------
     AGE|        0.0    .....(Fixed Parameter).....
    EDUC|     .00121***      .00025     4.84  .0000      .00072    .00170
 MARRIED|        0.0    .....(Fixed Parameter).....
  INCOME|     .00695***      .00148     4.69  .0000      .00404    .00985
  HHKIDS|     .00057***      .00019     3.02  .0026      .00020    .00095
--------+-------------------------------------------------------------------
+--------------------------------------------------------+
| Analysis of dummy variables in the model. The effects are |
| computed using E[y1|y2=1,d=1] - E[y1|y2=1,d=0] where d is |
| the variable. Variances use the delta method.  The effect |
| accounts for all appearances of the variable in the model.|
+--------------------------------------------------------+
|Variable      Effect   Standard error     t ratio         |
+--------------------------------------------------------+
 MARRIED      .009164     .006225            1.472
 HHKIDS       .000579     .000193            3.001
```

R11.5.7 Partial Effects in a Model with Direct and Indirect Effects

Lastly, there are models in which the effects can be identified as being ‘direct’ or ‘indirect.’ The basic sample selection model is a leading case. The model is a two equation structure,

$$d = 1[\boldsymbol{\gamma}'\mathbf{z} + u > 0]$$

$$y = \boldsymbol{\beta}'\mathbf{x} + \varepsilon$$

$$E[y|\mathbf{x},\mathbf{z},d=1] = \boldsymbol{\beta}'\mathbf{x} + (\rho\sigma)\lambda(\boldsymbol{\gamma}'\mathbf{z}).$$

where $\lambda(\boldsymbol{\gamma}'\mathbf{z})$ is based on the probability of selection into the sample. In this case, the direct partial effect of x on the regression part and the indirect partial effect on the probability part might be of separate interest. The reported effects would appear as in the example below:

```
-----------------------------------------------------------------------------
Sample Selection Model...........................
Two step     least squares regression ............
LHS=DOCVIS   Mean                   =       2.99463
Correlation of disturbance in regression
and Selection Criterion (Rho)...........    -.75369
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
  DOCVIS|  Coefficient      Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|   -1.64343       1.04388    -1.57  .1154    -3.68940    .40254
     AGE|     .06510***     .00768     8.47  .0000      .05004    .08016
    EDUC|     .26222**      .10182     2.58  .0100      .06265    .46178
 MARRIED|    -.32185        .20265    -1.59  .1122     -.71904    .07534
  LAMBDA|   -4.29958***    1.23383    -3.48  .0005    -6.71784  -1.88132
--------+--------------------------------------------------------------------
Partial effects of E[y] = Xb + c*L   with respect to the vector of
characteristics. They are computed at the means of the Xs. Means for direct
effects are for selected observations. Means for indirect effects are the
full sample used for the probit. If a variable appears in both Xb and in L
the second effect shown in the table is b + c*dL/dx = direct+indirect.
--------+--------------------------------------------------------------------
        |     Partial     Standard            Prob.      95% Confidence
  DOCVIS|      Effect       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Direct effects in the regression
     AGE|     .06510***     .00768     8.47  .0000      .05004    .08016
    EDUC|     .26222**      .10182     2.58  .0100      .06265    .46178
 MARRIED|    -.32185        .20265    -1.59  .1122     -.71904    .07534
        |Indirect effects in LAMBDA (means are for all obs.)
    EDUC|    -.22298***     .07472    -2.98  .0028     -.36943   -.07653
  HHNINC|   -1.42584***     .45069    -3.16  .0016    -2.30918   -.54251
  HHKIDS|    -.02788        .07490     -.37  .7097     -.17469    .11892
        |Total effect for variables in both parts
    EDUC|     .03924        .12630      .31  .7560     -.20830    .28677
--------+--------------------------------------------------------------------
```

R12: Model Predictions, Residuals, Simulations and Decompositions

R12.1 Introduction

This chapter will describe using the estimated model for prediction of the dependent variable. This involves several possible exercises. Section R12.2 discusses using the estimated model in the natural fashion to obtain fitted values for the dependent variable based on the data used to fit the model. Chapter R11 described analyses of scenarios for computing partial effects based on the last model estimated. Sections R12.3 and R12.4 continue that analysis by demonstrating how to use the estimated model to make predictions of the dependent variable under assumptions about the independent variable. For example, you might examine how the average prediction of a wage equation differs between men and women. Finally, Section R12.5 extends the analysis of the estimated model by showing how to compute the Oaxaca decomposition of differences in model predictions across two groups.

R12.2 Creating and Displaying Predictions and Residuals

Many of the single equation models in *LIMDEP*, though not all, contain a natural 'dependent variable.' Model predictions for any such model are easily obtained as discussed below. What constitutes a residual in these settings is ambiguous, but, once again, some construction that typically reflects a deviation of an actual from a predicted value can often be retained. The exact definition of a 'fitted value' and a 'residual' are given with the model descriptions in the *Econometric Modeling Guide*. The benchmark case is the linear regression model,

$$y_i = \boldsymbol{\beta}'\mathbf{x}_i + \varepsilon_i$$

for which the natural prediction is $\hat{y}_i = \mathbf{b}'\mathbf{x}_i$ and the residual is $e_i = y_i - \hat{y}_i$, where **b** is the estimated parameter vector. In many other cases, the predictor is only one possible function. For example, for the binary logit model, Prob($y_i = 1|\mathbf{x}_i$) = $\Lambda(\boldsymbol{\beta}'\mathbf{x}_i)$, one possible predictor is the conditional mean function, which is the probability, while another is $\hat{y}_i = 1$ if $\Lambda(\mathbf{b}'\mathbf{x}_i) > 0.5$ (or some other chosen value) and 0 otherwise. In either case, there is no obvious function to call the residual. As noted, when it is possible to define a candidate for predicting the dependent variable, you can save predictions after estimation.

There are several options for computing and saving fitted values from the regression models. You may request fitted values and/or residuals for almost any model. (The usual exceptions are, e.g., multiple equation models.) The fitted values are requested by adding

; Keep = name

to your model command. The request for residuals is

; Res = name

In each of these cases, the command will overwrite the variable if it already exists, or create a new one. In any model command, the following specification requests a listing of the residuals and several other variables:

; List

TIP: To keep fitted values in a text file, you can either use **; List** with an output file or use **WRITE** and write the values in their own file or **LIST ; variable $**.

If the current sample is not the entire data set, and the data array contains observations on the regressors but not the dependent variable, you can produce predicted values for these out of sample observations by adding the specification

; Fill

to your model command. The specifications **; Res**, **; Keep**, and **; Fill** do not compute values for any observations for which any variable to be used in the calculation is missing (i.e., equals -999). Otherwise, a prediction is computed for every row for which data can be found.

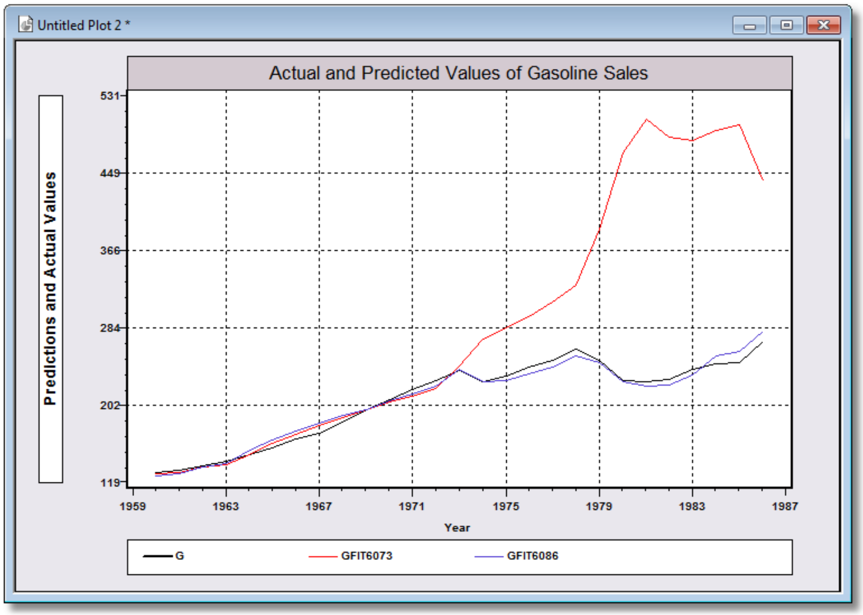
TIP: The specification **; Fill** provides a very simple way of generating out of sample predictions.

To provide an example of the **; Fill** feature, we will examine some data on gasoline sales in the U.S. before and after the 1973-1974 oil embargo. The data below are yearly series on gasoline sales (*g*), per capita income (*y*), and index numbers for a number of prices: *pg* is the gasoline price, *pnc*, *puc*, and *ppt* are price indices for new and used cars and public transportation, and *pn*, *pd*, and *ps* are aggregate price indices for nondurables, durables, and services.

IMPORT $
```
year,g,pg,y,pnc,puc,ppt,pd,pn,ps
1960 129.7   .925  6036  1.045   .836   .810   .444   .331   .302
1961 131.3   .914  6113  1.045   .869   .846   .448   .335   .307
1962 137.1   .919  6271  1.041   .948   .874   .457   .338   .314
1963 141.6   .918  6378  1.035   .960   .885   .463   .343   .320
1964 148.8   .914  6727  1.032  1.001   .901   .470   .347   .325
1965 155.9   .949  7027  1.009   .994   .919   .471   .353   .332
1966 164.9   .970  7280   .991   .970   .952   .475   .366   .342
1967 171.0  1.000  7513  1.000  1.000  1.000   .483   .375   .353
1968 183.4  1.014  7728  1.028  1.028  1.046   .501   .390   .368
1969 195.8  1.047  7891  1.044  1.031  1.127   .514   .409   .386
1970 207.4  1.056  8134  1.076  1.043  1.285   .527   .427   .407
1971 218.3  1.063  8322  1.120  1.102  1.377   .547   .442   .431
1972 226.8  1.076  8562  1.110  1.105  1.434   .555   .458   .451
1973 237.9  1.181  9042  1.111  1.176  1.448   .566   .497   .474
1974 225.8  1.599  8867  1.175  1.226  1.480   .604   .572   .513
1975 232.4  1.708  8944  1.276  1.464  1.586   .659   .615   .556
1976 241.7  1.779  9175  1.357  1.679  1.742   .695   .638   .598
1977 249.2  1.882  9381  1.429  1.828  1.824   .727   .671   .648
1978 261.3  1.963  9735  1.538  1.865  1.878   .769   .719   .698
1979 248.9  2.656  9829  1.660  2.010  2.003   .821   .800   .756
1980 226.8  3.691  9722  1.793  2.081  2.516   .892   .894   .839
1981 225.6  4.109  9769  1.902  2.569  3.120   .957   .969   .926
1982 228.8  3.894  9725  1.976  2.964  3.460  1.000  1.000  1.000
1983 239.6  3.764  9930  2.026  3.297  3.626  1.041  1.021  1.062
1984 244.7  3.707 10421  2.085  3.757  3.852  1.038  1.050  1.117
1985 245.8  3.738 10563  2.152  3.797  4.028  1.045  1.075  1.173
1986 269.4  2.921 10780  2.240  3.632  4.264  1.053  1.069  1.224
```

We will compute simple regressions of *g* on *one*, *pg*, and *y*. The first regression is based on the pre-embargo data, 1960-1973, but fitted values are produced for all 27 years. The second regression uses the full data set and also produces predicted values for the full sample. We then plot the actual and both predicted series on the same figure to examine the influence of the later data points.

```
DATE          ; 1960 $
PERIOD        ; 1960 - 1973 $
REGRESS       ; Lhs = g ; Rhs = one, pg, y ; Keep = gfit6073 ; Fill $
PERIOD        ; 1960 - 1986 $
REGRESS       ; Lhs = g ; Rhs = one, pg, y ; Keep = gfit6086  $
PLOT          ; Rhs = g, gfit6073, gfit6086 ; Grid
              ; Title = Actual and Predicted Values of Gasoline Sales
              ; Vaxis = Predictions and Actual Values $
```



The *Econometric Modeling Guide* will detail the formulas used in computing predictions, residuals, and accompanying information. When you use **; List**, some additional information will be displayed in your output. In some cases, there is no natural residual or prediction to be computed, for example in the bivariate probit model. In these cases, an alternative computation is done, so what is requested by **; Res** or **; Keep** may not actually be a residual or a fitted value. Individual model descriptions will provide details. In general, the **; List** specification produces the following:

1. An indicator of whether the observation was used in estimating the model. If not, the observation is marked with an asterisk,
2. The observation number or date if the data are time series,
3. The observed dependent variable when this is well defined,
4. The 'fitted value' = variable retained by **; Keep**,
5. The 'residual' = variable retained by **; Res**,
6. 'variable 1,' a useful additional function of the model which is not kept, and
7. 'variable 2,' another computation.

Although the last two variables are not kept internally, they are written to your output window and to the output file if one is open, so you can retrieve them later by editing the file with a word processor. In all cases, the formulas for these variables will be given, so if you need to have them at the time they are computed, you can use a subsequent **CREATE** command to obtain the variables.

We illustrate these computations with a Poisson regression and with the out of sample predictions generated by the regression above. The **POISSON** command would be

POISSON ; Lhs = … ; Rhs = … ; List $

The following table results and items listed for the Poisson model are:

Actual:	y	Prediction:	$E[y] = \exp(\mathbf{b'x})$
Residual:	$y - E[y]$	Index:	$\mathbf{b'x}$
Probability:	$\Pr[Y = y] = \exp(-\lambda)\lambda^y/y!$, $\lambda = E[y]$		

```
Predicted Values          (* => observation was not in estimating sample.)
Observation         Observed Y   Predicted Y   Residual        x(i)b     Pr[Y*=y]
         7          1.0000000    1.9798297   -.9798297     .6830109     .2734001
        11          3.0000000     .5306525   2.4693475    -.6336479     .0146494
        14          2.0000000     .2027126   1.7972874   -1.5959661     .0167762
        19          1.0000000     .5892748    .4107252    -.5288626     .3268881
        22            .000000    1.5780671  -1.5780671     .4562007     .2063736
        27            .000000    2.7696027  -2.7696027    1.0187039     .0626869
        30            .000000    2.7215365  -2.7215365    1.0011966     .0657736
        32            .000000    2.1277174  -2.1277174     .7550497     .1191089
        36            .000000    3.1435090  -3.1435090    1.1453397     .0431312
        40            .000000     .2942943   -.2942943   -1.2231751     .7450572
```

For a linear regression, the listed items are the familiar ones:

PERIOD ; 1960 - 1973 $
REGRESS ; Lhs = g ; Rhs = one, pg, y ; Keep = gfit6073 ; List ; Fill $

```
Predicted Values          (* => observation was not in estimating sample.)
Observation         Observed Y   Predicted Y   Residual     95% Forecast Interval
  1960              129.70000    127.97574    1.7242571    113.55855    142.39293
  1961              131.30000    129.46848    1.8315196    115.86233    143.07464
  1962              137.10000    134.79680    2.3032016    121.53562    148.05797
  1963              141.60000    138.04485    3.5551485    124.97181    151.11789
  1964              148.80000    148.57831     .2216922    134.92510    162.23151
  1965              155.90000    160.79099   -4.8909945    147.75083    173.83116
  1966              164.90000    170.39062   -5.4906219    157.42443    183.35681
  1967              171.00000    180.10723   -9.1072289    167.41426    192.80019
  1968              183.40000    187.94904   -4.5490359    175.14474    200.75334
  1969              195.80000    195.73348     .0665187    182.92095    208.54601
  1970              207.40000    204.03594    3.3640592    191.12778    216.94410
  1971              218.30000    210.46229    7.8377141    197.31386    223.61072
  1972              226.80000    219.00002    7.7999808    205.47309    232.52695
  1973              237.90000    242.56621   -4.6662105    226.75788    258.37454
* 1974              225.80000    271.46293   -45.662927    197.85242    345.07343
* 1975              232.40000    282.81499   -50.414993    193.97954    371.65044
* 1976              241.70000    295.83796   -54.137957    199.16406    392.51186
* 1977              249.20000    310.71199   -61.511988    201.13776    420.28621
* 1978              261.30000    328.38475   -67.084753    210.89623    445.87328
* 1979              248.90000    388.24798   -139.34798    168.33937    608.15660
* 1980              226.80000    469.95448   -243.15448    93.674489    846.23447
* 1981              225.60000    505.76061   -280.16061    67.359617    944.16161
* 1982              228.80000    486.72655   -257.92655    80.042437    893.41066
* 1983              239.60000    482.42593   -242.82593    97.683078    867.16878
* 1984              244.70000    493.02438   -248.32438    122.68188    863.36689
* 1985              245.80000    499.99091   -254.19091    126.69249    873.28934
* 1986              269.40000    439.61912   -170.21912    191.30350    687.93473
```

R12.3 The Last Model

The model used for the **PARTIAL EFFECTS** operation described in Chapter R11 is the last one that you estimated. This will be obvious from the results in your output window, though it is necessary to be specific about which function is being used. In nearly all cases, that function is a predictor for the dependent variable. Section R12.4 shows how to use that function to produce predictions for the model and analyze different scenarios, similar to the partial effects analysis in Chapter R11. In this case, the operation will be to **SIMULATE** the dependent variable.

For example, consider a logit model, fit with

```
LOGIT          ; Lhs = doctor
               ; Rhs = one,age,age^2,sex,income,sex*income $
```

The function used for the simulations will the logit probability,

$$\text{Last model} = \Lambda(\beta'x) = \text{Prob(Lhs variable} = 1).$$

which is the conditional mean function. There is a specific function used for each model for which you can use **PARTIAL EFFECTS** and **SIMULATE**. These are documented in the *Econometric Modeling Guide* for the particular models. At any time, you can find out what function is being used for the **SIMULATE** command by using the command

LAST MODEL $

For our logit example, the response would be

```
--> LAST MODEL $
The last estimated model is Logit Probability Function
```

In most cases, the function used is the conditional mean function. But, in some cases, such as the ordered probit or logit models, there a numerous probability functions. For this particular case, the default function is the probability of the highest outcome, for example,

```
--> OPROBIT ; Lhs = hsat ; Rhs=one,age,educ $
--> LAST MODEL $
The last estimated model is Ordered Probit     Probability Y =10
```

The ordered probit models are a special case. The highest category is usually the one of interest. You can change this by using

; Outcome = value (0,1,…)

NOTE: There is a default function for each model that **SIMULATE** may be used with. However, you can specify a different function to be analyzed. The alternative function need not even be a model. It can be any function you can specify with the command language. **SIMULATE** can analyze any variable in any function that is computed using data and parameters. Section R12.4.4 describes how to supply your own function to be analyzed.

R12.4 Using SIMULATE with the Last Model

Use **SIMULATE** after estimating a model to compute the average prediction of the dependent variable (or the average function value for the function saved by the last model). For example, the following fits a logit model, then computes the average predicted probability:

```
NAMELIST    ; xprobit = one,age,educ,income,income^2,age*income,hsat $
PROBIT      ; Lhs = doctor ; Rhs = xprobit $
```

```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               DOCTOR
Log likelihood function     -2727.43478
Restricted log likelihood   -2908.96085
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|     .86475***      .23176     3.73  .0002      .41050   1.31899
     AGE|     .01642***      .00422     3.89  .0001      .00815    .02469
    EDUC|    -.00166         .00872     -.19  .8494    -.01875    .01544
  INCOME|     .83966         .59137     1.42  .1557    -.31941   1.99874
        |Constructed variable INCOME^2.0
_ntrct01|    -.06449         .25065     -.26  .7970    -.55576    .42678
        |Interaction AGE*INCOME
_ntrct02|    -.02484**       .01176    -2.11  .0346    -.04789   -.00179
    HSAT|    -.15719***      .00985   -15.95  .0000    -.17650   -.13787
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
SIMULATE $
```

```
---------------------------------------------------------------------
Model Simulation Analysis for Probit Probability Function
---------------------------------------------------------------------
Simulations are computed by average over sample observations
---------------------------------------------------------------------
User Function      Function   Standard
(Delta method)      Value      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
Avg. function       .64718     .00686   94.33      .63373      .66062
```

Like **PARTIAL EFFECTS**, simulations are computed by averaging over the sample observations. You can, instead, compute the simulated value at the means of the data by adding

```
; Means
```

To the **SIMULATE** command. Weights may be used with

```
; Wts = the weighting variable
```

See Section R11.4.12 for further details of weighted observations in simulations.

R12.4.1 The Sample Used in the Simulation

You may use the estimation sample or any other defined sample for the simulations. After the last model is estimated, you may proceed immediately to the simulations using the estimation sample, or change the sample in any way with **SAMPLE**, **REJECT** or **INCLUDE**. The sample used for estimating the model need not be related to the sample used for the simulation.

You can save the results of the simulation by adding

```
                    ; Keep
```

to the **SIMULATE** command. This will create two variables in your data set, *function* and *se_fnctn* that contain the predictions and estimated standard errors of the predictions for the observations in the current sample. (Note, once again, the current sample need not be the one used to fit the model.) For example,

```
INCLUDE     ; New ; female = 1 $
PROBIT      ; Lhs = doctor ; Rhs = x $
INCLUDE     ; New ; female = 0 $
SIMULATE    ; Keep $
```

fits the model using the observations for which female equals one, then simulates the model for the observations for which female equals zero. The fitted values for the male half of the sample are generated using the coefficients computed with the female half of the sample.

R12.4.2 Scenarios in Simulations

The **SIMULATE** command operates the same way that **PARTIAL EFFECTS** does. The command used to examine different scenarios is

```
SIMULATE    ; Scenario
                 | variable   = list of values        and/or
                 & variable = range of values      and/or
                 @ variable = set of discrete values $
```

To continue our earlier example, we will simulate the average probability for ages 20, 25, 30, …, 80.

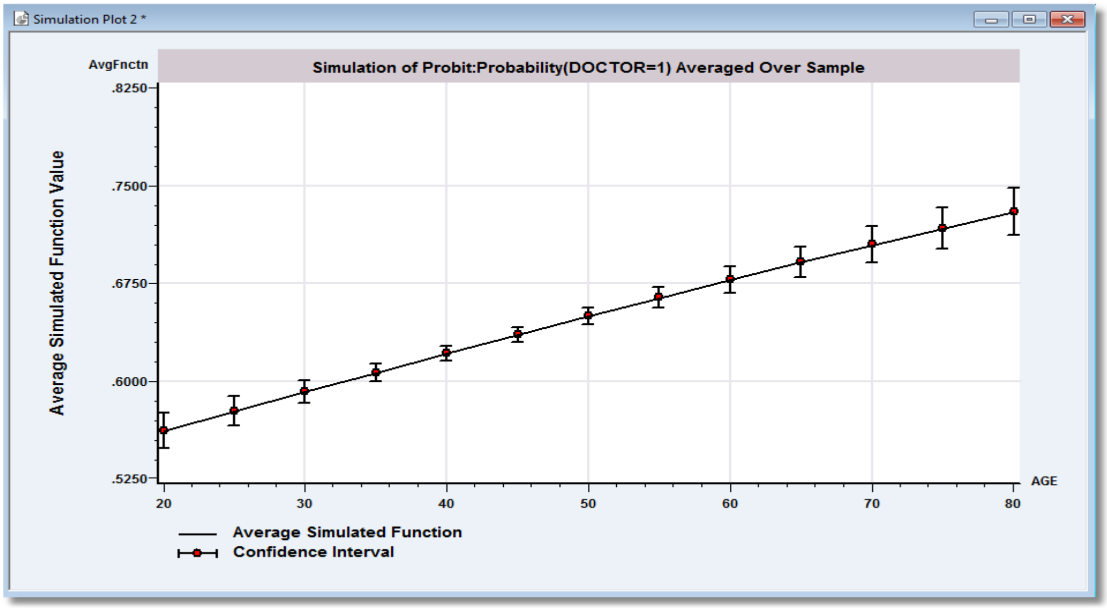
```
SIMULATE    ; Scenario & age = 20 (5) 80 $
```

```
----------------------------------------------------------------------
Model Simulation Analysis for Probit Probability Function
----------------------------------------------------------------------
Simulations are computed by average over sample observations
----------------------------------------------------------------------
User Function       Function   Standard
(Delta method)       Value      Error     |t|  95% Confidence Interval
----------------------------------------------------------------------
Avrg. Function       .62911     .00278  226.32      .62366      .63456
AGE      = 20.00     .56141     .00694   80.84      .54780      .57502
AGE      = 25.00     .57645     .00569  101.34      .56530      .58760
AGE      = 30.00     .59139     .00453  130.55      .58251      .60026
AGE      = 35.00     .60618     .00356  170.48      .59921      .61315
AGE      = 40.00     .62082     .00293  211.80      .61507      .62656
AGE      = 45.00     .63527     .00286  222.14      .62967      .64088
AGE      = 50.00     .64952     .00334  194.44      .64298      .65607
AGE      = 55.00     .66355     .00415  159.79      .65541      .67169
AGE      = 60.00     .67733     .00511  132.56      .66732      .68735
AGE      = 65.00     .69085     .00612  112.97      .67887      .70284
AGE      = 70.00     .70410     .00712   98.85      .69014      .71806
AGE      = 75.00     .71705     .00811   88.44      .70115      .73294
AGE      = 80.00     .72969     .00906   80.57      .71193      .74744
```

The settings for the scenarios are defined in Sections R11.4.5 and R11.4.8. You may also use

; Plot, ; Plot(ci) and ; Plot (c)

as described in Section R11.4.6. The preceding scenario shows the effect of age on the average probability graphically.



Scenarios may also set the values of particular variables, or change the values of particular variables. The specification is

; Set: settings separated by comma.

For example, the following command does a simulation of a conditional mean while advancing every person's age by 20 years and assuming all people are married:

```
NAMELIST    ; xdoc = one,age,educ,married $
POISSON     ; Lhs = docvis ; Rhs = xdoc $
SIMULATE    ; Set: age += 20 , married = 1 $
```

```
----------------------------------------------------------------------
Simulation and partial effects are computed with fixed settings
AGE      = change by +     =     20.0000
MARRIED  =                        1.0000
Simulation is computed with fixed settings and discrete changes
----------------------------------------------------------------------
Model Simulation Analysis for Exponential Regression Function
----------------------------------------------------------------------
Simulations are computed by average over sample observations
Value reported is change in function after the scenario is run.
----------------------------------------------------------------------
User Function      Function   Standard
(Delta method)      Value      Error     |t|  95% Confidence Interval
----------------------------------------------------------------------
Avrg. Function     1.70344     .01210  140.82     1.67973     1.72715
```

The scenario may set any variable (whether it is actually in the model or not) to a specific value, or to a changed value with one of

```
Variable *  = value
Variable +  = value
Variable -  = value
```

R12.4.3 Difference in Differences Analysis

There is a limitless variety of models and specifications for examining treatment effects – the contemporary literature is vast. There is little way to obtain generality, however, one particular approach is used reasonably often. Let pre- and post-treatment periods be denoted $t = 0$ and $t = 1$ and let the treated individuals be denoted '*T*' and controls be denoted '*C*.' Let the outcome variable be $y(t,T)$ or $y(t,C)$ – the outcome is generated by the conditional mean function of any model you specify, or a particular function that you specify such as in our examples above. We assume that there are three dummy variables that play some role in the model, *post_t* = 1(t = 1 and T), *post_c* = 1(t = 1 and C) and *pre_t* = 1(t = 0 and T). Pre-treatment controls are the base case. We assume that these dummy variables appear in the model somewhere so that treatment and status impact the outcome variable. The **SIMULATE** command can now be modified with

```
; did = post_t, post_c, pre_t
```

to produce an analysis of the difference in differences of the average outcomes. All other features of the **SIMULATE** command can be used with this extension.

To continue the earlier example, we construct a purely fictitious set of treatment and status variables and examine the difference in differences at ages 25, 24, 45 and 55.

```
CREATE      ; pret = Rnd(2) - 1 ; post = Rnd(2) - 1 ; postc = Rnd(2) - 1 $
POISSON     ; Lhs = hospvis ; Rhs = x,postt,postc,pret $
SIMULATE    ; did = postt,postc,pret ; Scenario: & age = 25(10)55 $
```

```
--------------------------------------------------------------------------------
Simulation is computing difference in difference
Specified settings for simulation
POSTT    = post effect treatment dummy
POSTC    = post effect controls  dummy
PRET     = pre  effect treatment dummy
Base category is pre-effect controls
D-i-D result is
{E[outcome|post treatment - E[outcome|post controls]} -
{E[outcome|pre  treatment - E[outcome|pre  controls]}
--------------------------------------------------------------------------------
Model Simulation Analysis for Exponential Regression Function
--------------------------------------------------------------------------------
Simulations are computed by average over sample observations
--------------------------------------------------------------------------------
User Function       Function   Standard
(Delta method)       Value      Error     |t|  95% Confidence Interval
-----------------------------------------------------------------------
Avrg. Function      -.02053     .01794    1.14     -.05569      .01464
AGE      = 25.00    -.16078     .19954     .81     -.55189      .23033
AGE      = 35.00    -.07021     .07400     .95     -.21525      .07483
AGE      = 45.00    -.03066     .02831    1.08     -.08614      .02482
AGE      = 55.00    -.01339     .01171    1.14     -.03634      .00956
--------------------------------------------------------------------------------
```

R12.4.4 Defining the Model for SIMULATE

The function that *LIMDEP* uses for **SIMULATE** is the model left behind by the previous model command. The model will remain in place until another fitted model changes its place. However, you can specify your own model, or function – it need not be a model; this can be any function that you wish to analyze. The additional information in the command is

; Function = any user defined function
; Covariance = matrix
; Parameters = set of values
; Labels = names of parameters

This is the same as described for **PARTIAL EFFECTS** in Section R11.4.14. The function definition is any function that you wish to specify using the same form as **MAXIMIZE**, **NLSQ**, **WALD**, etc. The function is assumed to involve an estimated parameter vector for which you also have in hand an estimated covariance matrix. The labels are provided so that you can differentiate between parameters and all the other numeric entities that can appear in the function.

To illustrate, we examine the behavior of the hazard function suggested in Section R11.4.14. The model is

$$\text{Prob}(y = 1 \mid \mathbf{x}) = \Phi(\boldsymbol{\beta}'\mathbf{x}).$$

The hazard function is

$$h(\boldsymbol{\beta}'\mathbf{x}) = -\text{dln}\Phi(-\boldsymbol{\beta}'\mathbf{x})/\text{d}(\boldsymbol{\beta}'\mathbf{x}) = \phi(\boldsymbol{\beta}'\mathbf{x})/[1 - \Phi(\boldsymbol{\beta}'\mathbf{x})].$$

This is not a conditional mean function, but it might nonetheless be interesting. To continue our example, we will employ this template and simulate the hazard function for a probit model. The income variable in our model enters the function nonlinearly in several terms. Step 1 is definition and estimation of the model

```
NAMELIST    ; xprobit = one,age,educ,income,income^2,age*income,hsat $
PROBIT      ; Lhs = doctor ; Rhs = xprobit $
```

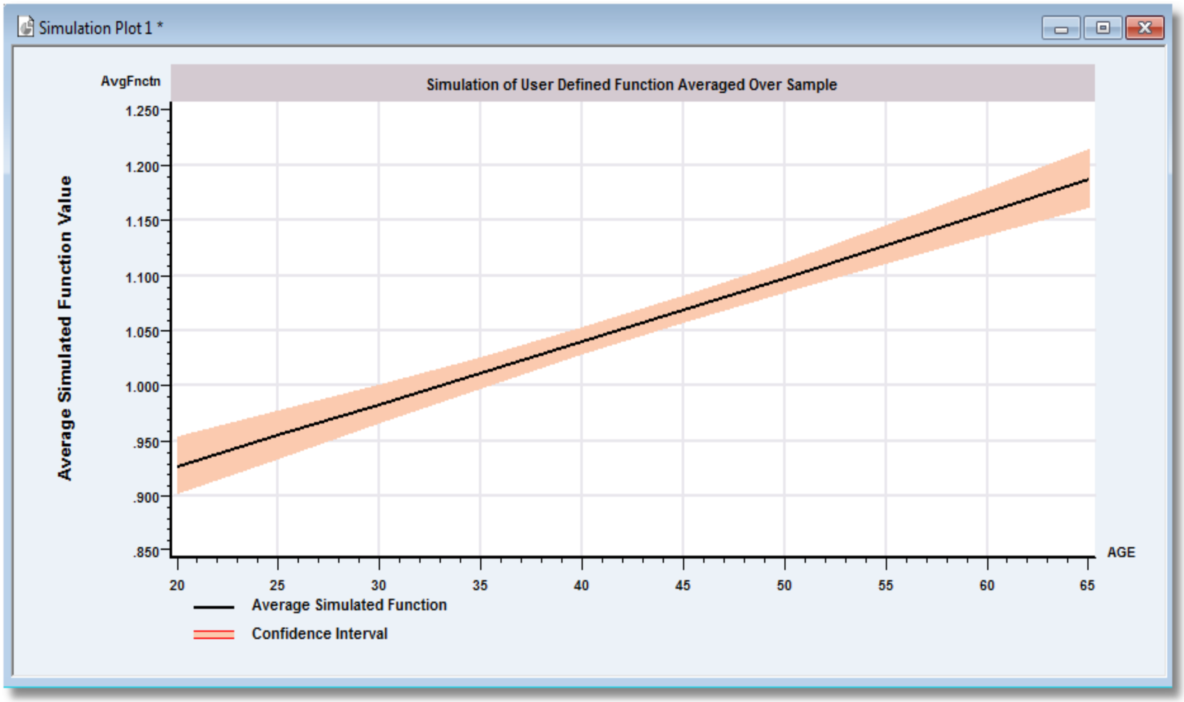
```
Normal exit:   4 iterations. Status=0, F=    2727.435
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               DOCTOR
Log likelihood function    -16635.39054
Restricted log likelihood  -18019.55173
Chi squared [  6](P= .000)   2768.32238
Significance level               .00000
McFadden Pseudo R-squared      .0768144
Estimation based on N =  27326, K =   7
Inf.Cr.AIC  =  33284.8 AIC/N =    1.218
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability......................................
Constant|    1.19157***      .09074    13.13  .0000     1.01372   1.36942
     AGE|     .01245***      .00162     7.70  .0000      .00928    .01562
    EDUC|    -.01481***      .00360    -4.11  .0000     -.02187   -.00775
  INCOME|     .46695**       .21382     2.18  .0290      .04788    .88602
        |Constructed variable INCOME^2.0
_ntrct01|     .03780         .06812      .55  .5789     -.09571    .17132
        |Interaction AGE*INCOME
_ntrct02|    -.01192***      .00441    -2.70  .0068     -.02055   -.00328
    HSAT|    -.17481***      .00396   -44.14  .0000     -.18257   -.16705
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Step 2 is the simulation.

```
SIMULATE    ; Labels = b1,b2,b3,b4,b5,b6,b7
            ; Parameters = b
            ; Covariance = varb
            ; Function = bx  = b1'xprobit |
                         cdf  = Phi(bx)    |
                         pdf = N01(bx)     |
                         pdf/(1-cdf)
            ; Scenario & age = 20(5)65
            ; Plot(ci) $
```

The function is defined recursively purely for convenience. The same results would be produced by **; Function = N01(b1’x)/(1 - Phi(b1’x))**; we decomposed it above to illustrate how to compute a complicated function in parts. The simulation and a plot with 95% confidence limits are as follows:

```
----------------------------------------------------------------------
Model Simulation Analysis for User Specified Function
----------------------------------------------------------------------
Simulations are computed by average over sample observations
----------------------------------------------------------------------
User Function       Function   Standard
(Delta method)       Value      Error      |t|  95% Confidence Interval
----------------------------------------------------------------------
Avrg. Function      1.06213     .00580  183.26     1.05077     1.07349
Base case has all category variables set to zero for all observs.
AGE      = 20.00     .92679     .01291   71.79      .90149      .95210
AGE      = 25.00     .95434     .01084   88.07      .93310      .97557
AGE      = 30.00     .98223     .00887  110.75      .96485      .99962
AGE      = 35.00    1.01048     .00716  141.06      .99644     1.02452
AGE      = 40.00    1.03906     .00602  172.54     1.02726     1.05086
AGE      = 45.00    1.06797     .00586  182.16     1.05648     1.07946
AGE      = 50.00    1.09720     .00682  160.95     1.08383     1.11056
AGE      = 55.00    1.12674     .00857  131.53     1.10995     1.14353
AGE      = 60.00    1.15658     .01077  107.44     1.13548     1.17768
AGE      = 65.00    1.18672     .01322   89.77     1.16081     1.21263
```


Simulation Plot 1 *
Simulation of User Defined Function Averaged Over Sample
AvgFnctn
Average Simulated Function Value
1.250
1.200
1.150
1.100
1.050
1.000
.950
.900
.850
20
25
30
35
40
45
50
55
60
65
AGE
Average Simulated Function
Confidence Interval

R12.4.5 Keeping the Simulation Results as a Variable

Average simulated values are shown above. You can also retain the underlying sample values by using

; Keep = the variable name

This will create a new variable (or replace an existing one) that contains the individual simulations for the observations used to create the averages. (Don't use this with **; Means** – all the values are the same.) When you keep the simulations, observations that are not used will be filled with missing values. This will also create an additional variable, *se_fnctn*, that contains the estimated standard errors using the delta method or Krinsky and Robb.

R12.4.6 The Distribution of the Simulated Values

Use

SIMULATE ; … ; quantiles $

to obtain a set of quantiles for the distribution of the simulated mean of the model or function. For example, the following shows estimates of the quantiles of the distribution of predictions from a probit model.

```
---------------------------------------------------------------------
Model Simulation Analysis for Probit:Probability(DOCTOR=1)
---------------------------------------------------------------------
Simulations are computed by average over sample observations
Results are saved as variable SIMKEEP . Std.Errors as SE_FNCTN
---------------------------------------------------------------------
User Function      Function   Standard
(Delta method)      Value      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
Avrg. Function      .63446     .00279  227.30      .62898      .63993

------------------------------------------------
Quantiles for Distribution of Simulated Function
------------------------------------------------
       Based on normal distribution
    Variance estimated by delta method
------------------------------------------------
         Quantile   Function Value
          .005             .62726
          .010             .62796
          .025             .62898
          .050             .62986
          .100             .63088
          .250             .63257
          .400             .63375
          .500             .63446
          .600             .63516
          .750             .63634
          .900             .63803
          .950             .63905
          .975             .63993
          .990             .64095
          .995             .64165
------------------------------------------------
```

R12.5 Oaxaca-Blinder Decompositions

The Oaxaca (1973) - Blinder (1973) decomposition is useful for examining the following situation: A model is fit for two groups (male/female, country A/country B, firm1/firm2, etc.). The average predictions of the two models are $\bar{y}_A$ and $\bar{y}_B$. We are interested in explaining the difference, $\bar{y}_A$ - $\bar{y}_B$. Much of the applicable literature is in labor economics, where the difference pertains to wage differences and the predictors are human capital variables such as age, education and experience. Consistent with the development in Section R12.4, write this difference as

$$\bar{y}_A - y_B = \bar{h}(b_A, x_A) - \bar{h}(b_B, x_B).$$

The question pursued by this technique is whether the difference is better explained, in general terms, by the difference between the coefficients, b_A - b_B or the difference between the covariates, x_A - x_B. In labor market applications such as Oaxaca's and Blinder's, the latter term is attributed to productivity and the residual is variously associated with labor market discrimination.

Most of the received literature on the decomposition focuses on the linear regression model. Our implementation of the method is general, and applies to any model that can be simulated using the **SIMULATE** command described in Section R12.4. The central idea behind the calculations is a term such as $\bar{h}(b_B, x_A)$ which is the mean outcome for group *A* if they had group *B*'s coefficients, and the reverse. For the Oaxaca-Blinder approach, the difference in mean outcomes can be written

$$\begin{aligned}\bar{y}_A - y_B &= \bar{h}(b_A, x_A) - \bar{h}(b_B, x_B)\\ &= \left[\bar{h}(b_A, x_A) - \bar{h}(b_A, x_B)\right] + \left[\bar{h}(b_A, x_B) - \bar{h}(b_B, x_B)\right].\end{aligned}$$

The first bracketed term is attributed to the difference in the data and the second is attributed to differences in the coefficients – we label these 'Due to data' and 'Due to beta' in the results of the procedure. The use of group *A* (with coefficients b_A) as the reference group for the decomposition is arbitrary, of course. Unfortunately, the decomposition is not crisp and symmetric even in the linear case. Several other approaches have been suggested. Daymont and Andrisani (1984) essentially reverse the roles of *A* and *B*, changing the viewpoint of the computation from *A* to B;

$$\begin{aligned}\bar{y}_A - y_B &= \bar{h}(b_A, x_A) - \bar{h}(b_B, x_B)\\ &= \left[\bar{h}(b_B, x_A) - \bar{h}(b_B, x_B)\right] + \left[\bar{h}(b_A, x_A) - \bar{h}(b_B, x_A)\right]\end{aligned}$$

A third approach, also proposed by Daymont and Andrisani (1984), suggests a three part decomposition that recognizes the possibility of an 'interaction' between coefficients and 'endowments,'

$$\begin{aligned}\bar{y}_A - y_B &= \bar{h}(b_A, x_A) - \bar{h}(b_B, x_B)\\ &= \left[\bar{h}(b_B, x_A) - \bar{h}(b_B, x_B)\right] + \left[\bar{h}(b_A, x_B) - \bar{h}(b_B, x_B)\right] +\\ &\quad + \left\{\left[\bar{h}(b_A, x_A) - \bar{h}(b_B, x_A)\right] - \left[\bar{h}(b_A, x_B) - \bar{h}(b_B, x_B)\right]\right\}\end{aligned}$$

In the linear model, the third part of this ‘three fold decomposition’ can be written as (b_A-b_B)(x_A-x_B). This neat construction does not carry over to nonlinear models. Finally, Oaxaca and Ransom (1994, 1998) and Neumark (1988) suggested basing the calculation on a common reference coefficient vector we’ll label b_*, so that

$$\begin{aligned}\bar{y}_A - y_B &= \bar{h}(b_A, x_A) - \bar{h}(b_B, x_B)\\ &= \left[\bar{h}(b_*, x_A) - \bar{h}(b_*, x_B)\right] + \\ &\quad \left[\bar{h}(b_A, x_A) - \bar{h}(b_*, x_A)\right] + \left[\bar{h}(b_*, x_B) - \bar{h}(b_B, x_B)\right].\end{aligned}$$

There is a pertinent question in this form of the model as to what should be used for the reference coefficient vector. Once again, attention focuses on the linear model, with various suggestions for a weighted average of b_A and b_B by Reimers (1983) (50% each), Cotton (1988) (the proportion of the full number of observations in each subsample), a matrix weighted average by Oaxaca and Ransom (1994) and the pooled estimator by Neumark (1988). We have adopted the last of these in our implementation in view of the idea that the usual application here will not be for the linear regression.

In order to compute the decompositions, it is necessary to fit the model three times, once with each subsample and then with the pooled sample. The syntax is

```
Model          ; For [variable = *,0,1] ; … the rest of the model $
DECOMPOSE $
```

The ‘**; For[..]**’ part of the model command specifies a discrete variable that takes at least the two values shown in your command. These need not be exhaustive, though typically the variable will be a dummy variable, such as *female* in our example below, that splits the full sample. The ‘*****’ indicates the pooled sample. For example, if your sample were grouped into five industries coded *industry* = 1,2,3,4,5 and you wished to compare industries 2 and 3, you could use

```
               ; For [industry = *,2,3] ; …
```

The model is any model that leaves behind a last model specification that can be used for the decomposition. In general, there are no other specifications for the **DECOMPOSE** command. In a few cases, it is necessary to provide an additional specification to complete the function definition. For example, if you fit an ordered probit model for *hsat* with, say, 11 categories coded 0,1,…,10, then

```
ORDERED        ; For [female = *,0,1] ; Lhs = hsat ; Rhs = … $
DECOMPOSE $
```

the decomposition would be applied to the probability for the last (*hsat* = 10) category. You could change this to the *hsat* = 8 category with

```
DECOMPOSE ; Outcome = 8 $
```

The other specific cases are the multinomial logit model and the bivariate probit models. These are noted in the applicable chapter in the *Econometric Modeling Guide*.

To illustrate, we start with a fairly involved probit model which we fit with the pooled sample, then we examine the average predicted probability for the two subsamples *female* = 0 and *female* = 1;

```
NAMELIST    ; xprobit = one,age,educ,income,income^2, age*income, hsat $
PROBIT      ; Lhs = doctor ; Rhs = xprobit $
SIMULATE    ; Scenario @ female = 0,1 $
```

The simulation seems to suggest that the average probabilities are essentially the same for the two groups. However, as we shall find, this masks a considerable underlying difference.

```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               DOCTOR
Log likelihood function     -2728.93368
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z     |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|     .86533***      .23148     3.74  .0002      .41163   1.31903
     AGE|     .01648***      .00421     3.91  .0001      .00822    .02474
    EDUC|    -.00218         .00871     -.25  .8024     -.01924    .01489
  INCOME|     .85493         .58931     1.45  .1469     -.30010   2.00996
        |Constructed variable INCOME^2.0
_ntrct01|    -.06908         .25004     -.28  .7823     -.55915    .42100
        |Interaction AGE*INCOME
_ntrct02|    -.02501**       .01175    -2.13  .0332     -.04803   -.00199
    HSAT|    -.15706***      .00985   -15.94  .0000     -.17637   -.13775
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
---------------------------------------------------------------------
Model Simulation Analysis for Probit Probability Function
---------------------------------------------------------------------
Simulations are computed by average over sample observations
---------------------------------------------------------------------
User Function      Function   Standard
(Delta method)      Value      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
Subsample for this iteration is FEMALE   =  0   Observations:    2313
Avg. function       .64008     .00705   90.83      .62627      .65390
Subsample for this iteration is FEMALE   =  1   Observations:    2170
Avg. function       .65460     .00690   94.92      .64109      .66812
```

The next set of instructions fits the models separately for the two groups and compares the three coefficient vectors. There does seem to be a substantive difference across the two groups. The two simulations based on separate coefficient vectors now show that the average predicted probabilities differ by about 20% (from 0.59 to 0.71).

```
INCLUDE    ; New ; year = 1988 & female = 0 $
PROBIT     ; Quiet ; Lhs = doctor ; Rhs = xprobit ; Table = Male $
SIMULATE    $
INCLUDE    ; New ; year = 1988 & female = 1 $
PROBIT     ; Quiet ; Lhs = doctor ; Rhs = xprobit ; Table = Female $
SIMULATE   $
INCLUDE    ; New ; year = 1988 $
PROBIT     ; Quiet ; Lhs = doctor ; Rhs = xprobit ; Table = Pooled $
REVIEW     ; Model = male, female, pooled ; Title = Estimated Probit Models $
```

```
+-----------------------------------------------------------------------------+
|                          Estimated Probit Models                            |
+----------+---------------------+---------------------+---------------------+
|          |        MALE         |       FEMALE        |       POOLED        |
+----------+---------------------+---------------------+---------------------+
| Variable |  Parameter|  t-ratio|  Parameter|  t-ratio|  Parameter|  t-ratio|
+----------+-----------+---------+-----------+---------+-----------+---------|
| Constant |      .8705|    2.810|      .6631|    1.844|      .8655|    3.739|
| AGE      |      .0161|    2.709|      .0147|    2.371|      .0165|    3.914|
| EDUC     |     -.0042|    -.361|      .0287|    2.016|     -.0022|    -.249|
| INCOME   |      .4734|     .611|     1.1573|    1.253|     .8548|    1.450|
| _ntrct01 |     -.1300|    -.370|     -.0821|    -.227|     -.0691|    -.276|
| _ntrct02 |     -.0157|    -.982|     -.0308|   -1.713|     -.0250|   -2.129|
| HSAT     |     -.1731|  -12.719|     -.1437|   -9.925|     -.1571|  -15.947|
+----------+-----------+---------+-----------+---------+-----------+---------+
---------------------------------------------------------------------
Model Simulation Analysis for Probit Probability Function
Simulations are computed by average over sample observations
---------------------------------------------------------------------
User Function      Function   Standard
(Delta method)      Value      Error     |t|  95% Confidence Interval
---------------------------------------------------------------------
Subsample for this iteration is FEMALE   =  0   Observations:    2311
Avg. function       .58558     .00974   60.10      .56648      .60467
Subsample for this iteration is FEMALE   =  1   Observations:    2170
Avg. function       .71256     .00943   75.52      .69407      .73105
```

Finally, the decomposition is produced by

```
PROBIT     ; For [female = *,0,1] ; Lhs = doctor ; Rhs = xprobit $
DECOMPOSE $
```

The results shown below decompose the difference by the several methods suggested earlier. As part of the output, a chi squared (Wald) test of the difference between the two coefficient vectors. In this application, the hypothesis that the two coefficient vectors are the same is decisively rejected. The decomposition analysis follows. It appears that for these data and this application, the large majority of the difference between the average predictions is explained by variation in the coefficients rather than variation in the data.

```
-------------------------------------------------------------------
Decomposition of Changes in Average Functions
Model Used in Computations = Probit Probability Function
-------------------------------------------------------------------
               Sample is FEMALE  = 0         FEMALE  = 1     Sample
Estimates Based on              (0)                 (1)      Size
FEMALE   = 0 (a)        .585578 (a,0)       .602840 (a,1)      2313
FEMALE   = 1 (b)        .709426 (b,0)       .712562 (b,1)      2170
Pooled   =** (*)        .640084 (*,0)       .654605 (*,1)      4483
-------------------------------------------------------------------
Wald Test of Difference in the Two Coefficient Vectors
Chi squared[  7] =   79.7955        , P Value =  .0000
-------------------------------------------------------------------
Total Change in Function    (a,0) - (b,1) =     -.126984 ( 100.00%)
-------------------------------------------------------------------
Oaxaca    | Due to data is  (a,0) - (a,1) =     -.017262 (  13.59%)
Blinder   | Due to beta is  (a,1) - (b,1) =     -.109723 (  86.41%)
-------------------------------------------------------------------
Daymont   | Due to data is  (b,0) - (b,1) =     -.003136 (   2.47%)
Andrisani | Due to beta is  (a,0) - (b,0) =     -.123848 (  97.53%)
-------------------------------------------------------------------
Daymont   | Due to data is  (b,0) - (b,1) =     -.003136 (   2.47%)
Andrisani | Due to beta is  (a,1) - (b,1) =     -.109723 (  86.41%)
(3 Fold)  | Due to function (a,0) - (b,0) +
          |                 (a,1) + (b,1) =     -.014126 (  11.12%)
-------------------------------------------------------------------
Ransom    | Due to data is  (*,0) - (*,1) =     -.014521 (  11.44%)
Oaxaca    | Due to beta is  (a,0) - (*,0) +     -.112463 (  88.56%)
Neumark   |                 (*,1) - (b,1)
-------------------------------------------------------------------
```

R13: Testing Hypotheses and Imposing Restrictions

R13.1 Introduction

This chapter describes procedures for testing hypotheses and imposing restrictions on estimated parameters. *LIMDEP* contains a wide variety of procedures which can be used for hypothesis testing, including the familiar trio of tests, Wald, LM, LR as well as Hausman tests and other moment based tests of model specification. Sections R13.2-R13.5 will describe how to carry out *F* tests in linear regressions and Wald, LM and LR tests in other models (including the linear regression). Hausman tests and conditional moment tests are described in Chapter R16. Models in *LIMDEP* may be estimated subject to linear restrictions. In most case, these will be used either to force coefficients to equal each other or to be fixed at specific values. The options described in Section R13.6 include these and more general linear restrictions. Tests of hypotheses that involve nonlinear functions of the parameters are described in Chapter R14.

R13.2 F Test of Linear Restrictions in Linear Models

The following applies only to the linear regression models, **REGRESS**, **2SLS, SURE, 3SLS** and the fixed effects linear model, **REGRESS ; Panel; Fixed Effects**. In the settings mentioned, the parameters of the model are estimated by least squares. This produces an 'unconstrained' least squares estimator. A restricted model can be conveniently estimated and tested in *LIMDEP* by specifying the regression as usual and adding the restrictions as an optional specification. For the linear models, the restrictions are specified by adding the following specification to the model command:

; CLS: linear function = value, linear function = value,...

For example, suppose the parameter vector has five elements, and three restrictions to be imposed are

$$\begin{aligned} b_2 &+ 2b_3 - b_4 = .5 \\ 3b_1 &+ 1.2b_2 - b_4 = 0 \\ b_4 &- b_5 = 0 \end{aligned}$$

The regression could be specified with

```
REGRESS     ; Lhs = y ; Rhs = x1,x2,x3,x4,x5
            ; CLS: b(2)   +    2b(3) - b(4) = .5,
                   3b(1) + 1.2b(2) - b(4) =  0,
                   b(4)   -      b(5)        =  0 $
```

The restrictions are written exactly as they appear in theory. Note, however, they must be written in the form 'linear function = value.' Also, *separate restrictions with commas, not semicolons.*

This specification, when used with **REGRESS**, **SURE**, **2SLS**, or **3SLS** produces a full set of output for the restricted estimator as well as for the unrestricted one (if it exists). The appropriate test statistic is also presented. When using the **; CLS:** option with one of the systems estimators, $s^2(\mathbf{X'X})^{-1}$ is replaced with the variance matrix of the GLS estimator. The parameter vector in these models is obtained by stacking the parameter vectors in the individual equations.

HINT: The results for **REGRESS**, **SURE**, **2SLS**, and **3SLS** which are automatically saved by this procedure, for example, matrices *b* and *varb*, are the *restricted* estimates.

NOTE: This procedure may be used for only one set of restrictions. If your command contains **; CLS:... ; CLS:...** for a second or more sets of restrictions, only the last one will be carried out. To analyze multiple sets of restrictions, the **; Test** form of the command described in Section R13.3 should be used. This will produce several test statistics, but not more than one restricted estimator.

Restrictions specified as shown in the previous example make it obvious exactly what the hypothesis is. However, there is a bit of inconvenience in that the indexes in the parameters change if the equation changes. For example, in the specification

```
REGRESS      ; Lhs = hhninc ; Rhs = one,age,educ,married,hhkids,health $
```

to force the *age* coefficient to equal 0 and the *married* and *hhkids* coefficients to equal each other, we would use b(2) = 0, b(4)-b(5) = 0. However, if another variable, say *female*, were added to the equation between *educ* and *married*, then the second constraint would have to be changed to b(5)-b(6) = 0. An alternative syntax may be used to remove this relationship. To impose the constraint, the equation can be specified as

```
REGRESS      ; Lhs = hhninc ; Rhs = one,age,educ,married,hhkids,health
             ; CLS:  age = 0, married - hhkids = 0 $
```

The obvious advantage of this syntax is that it is not tied to the position of the variables in the Rhs list. To continue our example, if *female* is now inserted into the equation between *educ* and *married*, the restrictions do not have to be changed. Coefficients in the restrictions are given as before, but now must include a '*' for multiplication. For example, the restriction above could be **married - 3.5*hhkids = 0**.

You may use either of these two syntaxes to specify restrictions in any single equation model. When there are multiple equations, there can be multiple appearances of each variable. As such, the syntax based on names will no longer work, and you must use the parameter index form. For example, in order to impose equality of the two *age* coefficients in

```
SURE         ; Lhs = income,educ ; Eq1 = one,age,married ; Eq2 = one,age,hhkids $
```

It would be necessary to use **; CLS: b(2) - b(5) = 0**. The alternative, **; CLS: age - age = 0** (while true) would not be useable

From this point forward, we will use the syntax based on variable names rather than parameter names whenever possible.

When you specify a constrained linear regression, the output will include the full unconstrained results and the constrained estimates. The constrained results will also contain the F statistic and a test of the restrictions as a hypothesis. The application below carries out the test suggested in the earlier example.

NOTE: The **; CLS** and **; Test** specifications described in the next section cannot detect interaction terms and nonlinear specifications in model commands. For example,

```
REGRESS      ; Lhs = y ; Rhs = one,x1,x2,x1*x2
             ; CLS: x1*x2 = 0 $
```

will produce an error message. In order to fit the restricted regression and test the hypothesis about the interaction term, it is necessary to create, say, *x*1*x*2 = *x*1***x*2, then operate directly on the single variable.

```
-----------------------------------------------------------------------------
Ordinary     least squares regression ............
LHS=HHNINC   Mean                 =          .34890
             Standard deviation   =          .16405
             No. of observations  =            4481  Degrees of freedom
Regression   Sum of Squares       =         14.8647           5
Residual     Sum of Squares       =         105.709        4475
Total        Sum of Squares       =         120.573        4480
             Standard error of e  =          .15369
Fit          R-squared            =          .12328  R-bar squared =   .12230
Model test   F[  5,  4475]        =       125.85407  Prob F > F*   =   .00000
Diagnostic   Log likelihood       =      2036.69639  Akaike I.C.   = -3.74424
             Restricted (b=0)     =      1741.91120
             Chi squared [  5]    =       589.57038  Prob C2 > C2* =   .00000
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  HHNINC|  Coefficient       Error        z     |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .06186***      .01882     3.29  .0010      .02497    .09876
     AGE|    -.00034         .00024    -1.44  .1487     -.00080    .00012
    EDUC|     .01943***      .00099    19.62  .0000      .01748    .02137
 MARRIED|     .09572***      .00587    16.30  .0000      .08421    .10724
  HHKIDS|    -.02954***      .00548    -5.39  .0000     -.04028   -.01880
  HEALTH|     .00250**       .00105     2.37  .0176      .00044    .00456
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
Restricted   least squares regression ............
LHS=HHNINC   Mean                 =          .34890
             Standard deviation   =          .16405
             No. of observations  =            4481  Degrees of freedom
Regression   Sum of Squares       =         10.1564           3
Residual     Sum of Squares       =         110.417        4477
Total        Sum of Squares       =         120.573        4480
             Standard error of e  =          .15705
Fit          R-squared            =          .08423  R-bar squared =   .08362
Model test   F[  3,  4477]        =       137.26823  Prob F > F*   =   .00000
Diagnostic   Log likelihood       =      1939.06322  Akaike I.C.   = -3.70155
             Restricted (b=0)     =      1741.91120
             Chi squared [  3]    =       394.30405  Prob C2 > C2* =   .00000
Restrictions F[  2,  4475] ←      =        99.65804  Prob F > F*   =   .00000
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  HHNINC|  Coefficient       Error        z     |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .09931***      .01383     7.18  .0000      .07220    .12641
     AGE|       .000    .....(Fixed Parameter).....
    EDUC|     .01823***      .00100    18.22  .0000      .01627    .02019
 MARRIED|     .02808***      .00320     8.78  .0000      .02181    .03435
  HHKIDS|     .02808***      .00320     8.78  .0000      .02181    .03435
  HEALTH|     .00125         .00105     1.19  .2341     -.00081    .00331
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
Fixed parameter ... is constrained to equal the value or
had a nonpositive st.error because of an earlier problem.
-----------------------------------------------------------------------------
```

R13.3 Testing Linear Restrictions Using the Wald Statistic

The Wald statistic for linear restrictions is based on the general linear hypothesis,

$$H_0\text{:}\ \ \mathbf{R\beta} - \mathbf{q} = \mathbf{0}\ \ \text{vs.}\ H_1\text{:}\ \ \mathbf{R\beta} - \mathbf{q} \neq \mathbf{0},$$

where $\boldsymbol{\beta}$ is a set of K coefficients that appear in the specified model, $\mathbf{R}$ is a $J\times K$ matrix of constants that specify the restrictions, and $\mathbf{q}$ is a $J\times 1$ vector of constants. (See, e.g., Greene (2012, Chapter 5) for discussion.) It is assumed that you are imposing fewer restrictions than there are parameters, so J is strictly less than K. *LIMDEP* provides several ways to compute linear functions of parameters and associated standard errors, and to compute test statistics for analyzing this form of restriction. The general specification

; Test: linear restrictions

can be used with all models that are fit with the program. The specification can be used to test a hypothesis involving one or more restrictions and one or more hypotheses at the same time. Individual restrictions within a hypothesis are separated by commas. Hypotheses are separated by the vertical bar character, '|.'

NOTE: In *LIMDEP* Version 9, the specification was **; Wald:**. This is still usable.

The restrictions are set up in exactly the same fashion as for **; CLS:**. The specification produces the Wald test statistic and the significance level in the output.

In all cases, including the linear regression model, the Wald statistic is computed using the results of the estimated model *without imposing restrictions*. The computed statistic is

$$Wald = \left(\mathbf{R}\hat{\boldsymbol{\beta}} - \mathbf{q}\right)'\left[\mathbf{R}\hat{\boldsymbol{\Sigma}}\mathbf{R}'\right]^{-1}\left(\mathbf{R}\hat{\boldsymbol{\beta}} - \mathbf{q}\right)$$

where $\hat{\boldsymbol{\beta}}$ is the estimated parameter vector and $\hat{\boldsymbol{\Sigma}}$ is the estimated covariance matrix. Under the assumptions of the model and the hypothesis, this is a chi squared statistic with degrees of freedom equal to the number of restrictions (which equals the number of rows in $\mathbf{R}$, or elements in $\mathbf{q}$).

NOTE: The *restricted* least squares or maximum likelihood estimator is not computed in order to obtain the Wald statistic. The Wald statistic is based on the unrestricted estimates. It can be seen in the examples below, the reported model estimates are the unrestricted estimates.

For an example, consider a translog production function

$$y = \beta_1 + \beta_2 x1 + \beta_3 x2 + \beta_4 x1^2 + \beta_5 x2^2 + \beta_6(2x1{*}x2) + \varepsilon,$$

where the variables are logs of output and the inputs. The hypothesis of constant returns to scale (CRTS) in this model involves two restrictions, $\beta_2+\beta_3 = 1$ and $\beta_4+\beta_5+\beta_6 = 0$. The hypothesis of the Cobb-Douglas model as a restriction on the translog model is $\beta_4 = 0$, $\beta_5 = 0$, $\beta_6 = 0$.

Testing for CRTS in the two model forms could be done as follows:

```
REGRESS    ; Lhs = y ; Rhs = one,x1,x2,x11,x22,x12
           ; Test:  x1 + x2 = 1, x11 + x22 + x12 = 0  |    ? CRTS translog
                    x1 + x2 = 1, x11 = 0, x22 = 0, x12 = 0 $ CRTS Cobb-Douglas
```

When your specification tests a single restriction, the test results are embedded in the model results. When you specify more than one hypothesis test, a separate table of results is produced. The example below demonstrates.

The data listed are statewide measures of inputs and output in the transportation sector from Greene (2012, Table F7.2). After importing the data, we created the transformed variables used in the production function

```
IMPORT $

State            ValueAdd     Capital     Labor         NFirm
Alabama           126.148       3.804    31.551            68
California       3201.486     185.446   452.844          1372
Connecticut       690.670      39.712   124.074           154
Florida            56.296       6.547    19.181           292
Georgia           304.531      11.530    45.534            71
Illinois          723.028      58.987    88.391           275
Indiana           992.169     112.884   148.530           260
Iowa               35.796       2.698     8.017            75
Kansas            494.515      10.360    86.189            76
Kentucky          124.948       5.213    12.000            31
Louisiana          73.328       3.763    15.900           115
Maine              29.467       1.967     6.470            81
Maryland          415.262      17.546    69.342           129
Massachusetts     241.530      15.347    39.416           172
Michigan         4079.554     435.105   490.384           568
Missouri          652.085      32.840    84.831           125
NewJersey         667.113      33.292    83.033           247
NewYork           940.430      72.974   190.094           461
Ohio             1611.899     157.978   259.916           363
Pennsylvania      617.579      34.324    98.152           233
Texas             527.413      22.736   109.728           308
Virginia          174.394       7.173    31.301            85
Washington        636.948      30.807    87.963           179
WestVirginia       22.700       1.543     4.063            15
Wisconsin         349.711      22.001    52.818           142

CREATE     ; x1 = Log(capital/nfirm) ; x2 = Log(labor/nfirm)
           ; x11 = x1*x1 ; x22 = x2*x2 ; x12 = 2*x1*x2 $
CREATE     ; y = Log(valueadd/nfirm) $
REGRESS    ; Lhs = y ; Rhs = one,x1,x2,x11,x22,x12
           ; Test: x1+ x2 = 1, x11 + x22 + x12 = 0 $
```

The first regression tests the single hypothesis of constant returns to scale in the translog model.

```
-----------------------------------------------------------------------------
Ordinary     least squares regression ............
LHS=Y        Mean                 =          .77173
             Standard deviation   =          .89931
             No. of observations  =              25  Degrees of freedom
Regression   Sum of Squares       =         18.8441           5
Residual     Sum of Squares       =         .565973          19
Total        Sum of Squares       =         19.4100          24
             Standard error of e  =          .17259
Fit          R-squared            =          .97084  R-bar squared =   .96317
Model test   F[  5,    19]        =       126.52092  Prob F > F*   =   .00000
Diagnostic   Log likelihood       =        11.87760  Akaike I.C.   = -3.30809
             Restricted (b=0)     =       -32.30989
             Chi squared [  5]    =        88.37497  Prob C2 > C2* =   .00000
Wald Test: Chi-Squared( 2) = 3.25567    Significance level =   .19635 <--
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       t    |t|>T*         Interval
--------+--------------------------------------------------------------------
Constant|    1.92982***       .17886    10.79  .0000     1.57925   2.28039
      X1|    -.02044          .29835     -.07  .9461     -.60520    .56431
      X2|     .66866          .42676     1.57  .1337     -.16778   1.50510
     X11|    -.07322          .12540     -.58  .5662     -.31900    .17256
     X22|     .02341          .19785      .12  .9070     -.36436    .41119
     X12|    -.03567          .15238     -.23  .8174     -.33434    .26300
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The second regression tests the CRTS hypothesis in the translog model then imposes the restrictions of the Cobb-Douglas model and tests for CRTS in the restricted model.

```
REGRESS    ; Lhs = y ; Rhs = one,x1,x2,x11,x22,x12
           ; Test:  x1+ x2 = 1, x11 + x22 + x12 = 0  |
                    x11 = 0, x22 = 0, x12 = 0 , x1 + x2 = 1 $
```

```
-----------------------------------------------------------------------------
Ordinary     least squares regression ............
LHS=Y        Mean                 =          .77173
             Standard deviation   =          .89931
             No. of observations  =              25  Degrees of freedom
Regression   Sum of Squares       =         18.8441           5
Residual     Sum of Squares       =         .565973          19
Total        Sum of Squares       =         19.4100          24
             Standard error of e  =          .17259
Fit          R-squared            =          .97084  R-bar squared =   .96317
Model test   F[  5,    19]        =       126.52092  Prob F > F*   =   .00000
Diagnostic   Log likelihood       =        11.87760  Akaike I.C.   = -3.30809
             Restricted (b=0)     =       -32.30989
             Chi squared [  5]    =        88.37497  Prob C2 > C2* =   .00000
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       t    |t|>T*         Interval
--------+--------------------------------------------------------------------
Constant|    1.92982***       .17886    10.79  .0000     1.57925   2.28039
      X1|    -.02044          .29835     -.07  .9461     -.60520    .56431
      X2|     .66866          .42676     1.57  .1337     -.16778   1.50510
     X11|    -.07322          .12540     -.58  .5662     -.31900    .17256
     X22|     .02341          .19785      .12  .9070     -.36436    .41119
     X12|    -.03567          .15238     -.23  .8174     -.33434    .26300
```

```
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Chi squared tests of linear restrictions. Degrees of freedom shown
in [.]. Equals zero is implied if no specific value was given.
 1. Restriction:X1+X2=1,X11+X22+X12=0
    Chi squared[ 2] =        3.256, P value =  .1964 ←
 2. Restriction:X11=0,X22=0,X12=0,X1+X2=1
    Chi squared[ 4] =       24.905, P value =  .0001
-----------------------------------------------------------------------------
```

The identical syntax is used for nonlinear models and the results will be arranged similarly. In the following example, the command first tests the joint hypothesis that all the coefficients in the model equal zero, then it tests each of the hypotheses one at a time.

```
POISSON   ; Lhs = docvis ; Rhs = one,age,educ,public,married,hhkids
          ; Test:  age = 0,  educ = 0,  public = 0,  married = 0,  hhkids = 0 |
                   age = 0 | educ = 0 | public = 0 | married = 0 | hhkids = 0 $
```

```
-----------------------------------------------------------------------------
Poisson Regression
Dependent variable                DOCVIS
Log likelihood function    -15891.44190
Restricted log likelihood  -16398.15386
Chi squared [    5 d.f.]     1013.42392 ←
Significance level               .00000
McFadden Pseudo R-squared      .0309005
Estimation based on N =   4481, K =   6
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCVIS|  Coefficient       Error       z     |z|>Z*        Interval
--------+--------------------------------------------------------------------
Constant|     .42184***      .08261     5.11  .0000      .25994    .58375
     AGE|     .01673***      .00089    18.75  .0000      .01498    .01848
    EDUC|   -.02491***      .00438    -5.68  .0000    -.03350   -.01631
  PUBLIC|     .28806***      .03258     8.84  .0000      .22421    .35191
 MARRIED|   -.03153          .02198    -1.43  .1515    -.07461    .01156
  HHKIDS|   -.20139***      .02217    -9.08  .0000    -.24484   -.15793
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Chi squared tests of linear restrictions. Degrees of freedom shown
in [.]. Equals zero is implied if no specific value was given.
 1. Restriction:AGE=0,EDUC=0,PUBLIC=0,MARRIED=0,HHKIDS=0
    Chi squared[ 5] =      992.331, P value =  .0000 ←
 2. Restriction:AGE=0
    Chi squared[ 1] =      351.491, P value =  .0000
 3. Restriction:EDUC=0
    Chi squared[ 1] =       32.272, P value =  .0000
 4. Restriction:PUBLIC=0
    Chi squared[ 1] =       78.181, P value =  .0000
 5. Restriction:MARRIED=0
    Chi squared[ 1] =        2.057, P value =  .1515
 6. Restriction:HHKIDS=0
    Chi squared[ 1] =       82.500, P value =  .0000
-----------------------------------------------------------------------------
```

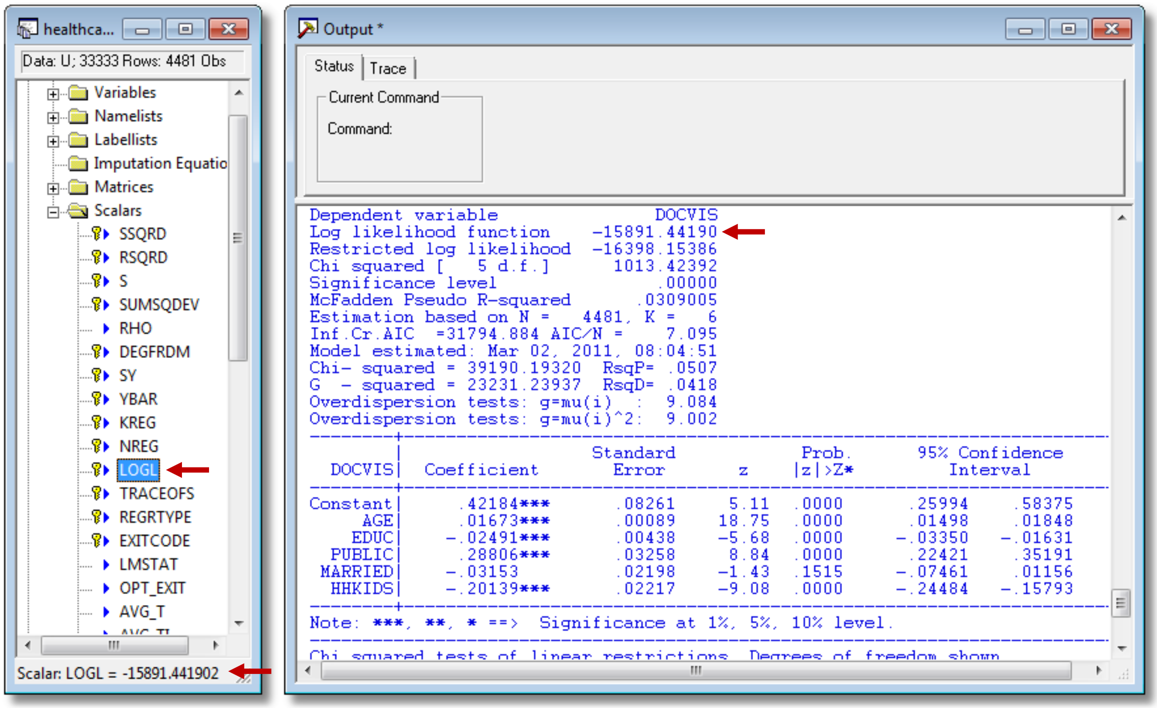
Note that there are two joint tests of the hypothesis that all coefficients are equal to zero in the results above. The Wald statistic is shown in restriction 1 in the lower table of results. The likelihood ratio test of the same hypothesis is shown with the standard results at the top of the table of results.

R13.4 Likelihood Ratio Tests

Most of the models in *LIMDEP* are estimated using the maximum likelihood estimator. The log likelihood function provides a general approach to testing hypotheses. In most cases, the general test of the hypothesis that all coefficients in the estimated model save for the constant term are equal to zero is part of the standard output. In the Poisson regression example at the end of the previous section, the initial model results include a likelihood ratio (chi squared) test of the hypothesis. The general result for the test is based on the likelihood ratio statistic,

$$LR = 2[\log L(\mathit{unrestricted}) - \log L(\mathit{restricted})].$$

Under the assumptions of the model and assuming the test is appropriate, the statistic is a chi squared statistic with degrees of freedom equal to the number of restrictions. *LIMDEP* automatically saves the log likelihood value in a scalar named *logl* when you fit a model. This value is replaced each time you fit a new model. Figure R13.1 illustrates this for the Poisson model fit in the previous example.



Scalar: LOGL = -15891.441902

```
Dependent variable                DOCVIS
Log likelihood function    -15891.44190
Restricted log likelihood  -16398.15386
Chi squared [   5 d.f.]      1013.42392
Significance level               .00000
McFadden Pseudo R-squared      .0309005
Estimation based on N =    4481, K =   6
Inf.Cr.AIC  =31794.884 AIC/N =    7.095
Model estimated: Mar 02, 2011, 08:04:51
Chi- squared = 39190.19320  RsqP= .0507
G  - squared = 23231.23937  RsqD= .0418
Overdispersion tests: g=mu(i)  :  9.084
Overdispersion tests: g=mu(i)^2:  9.002
--------+--------------------------------------------------------------------
        |                 Standard            Prob.        95% Confidence
  DOCVIS|  Coefficient      Error        z    |z|>Z*          Interval
--------+--------------------------------------------------------------------
Constant|      .42184***     .08261      5.11  .0000      .25994     .58375
     AGE|      .01673***     .00089     18.75  .0000      .01498     .01848
    EDUC|    -.02491***      .00438     -5.68  .0000    -.03350    -.01631
  PUBLIC|      .28806***     .03258      8.84  .0000      .22421     .35191
 MARRIED|    -.03153         .02198     -1.43  .1515    -.07461     .01156
  HHKIDS|    -.20139***      .02217     -9.08  .0000    -.24484    -.15793
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Chi squared tests of linear restrictions. Degrees of freedom shown
```

Figure 13.1 Project Window with Saved Log Likelihood

The general strategy for obtaining the statistic would be

```
Model     ; specification with restrictions … $
CALC      ; loglr = logl $
Model     ; specification without restrictions … $
CALC      ; loglu = logl $
```

Now, you can compute the likelihood ratio statistic. For example, the following will display the statistic and appropriate critical value from the chi squared table. (You must provide the degrees of freedom – there is no way for the program to figure out the degrees of freedom based on the commands or the results.)

```
CALC      ; List  ;  lrtest = 2*(loglu - loglr) ; Ctb(.95, … <df>…) $
```

The following will show several typical applications.

R13.4.1 Fixed Value Restriction in a Poisson Model

The following computes an unrestricted probit model, then tests the hypothesis that the coefficients on the last two variables are zero. The initial output is suppressed. Only the results of the test are shown.

```
PROBIT    ; Quiet
          ; Lhs = doctor ; Rhs = one,age,educ,public,married,hhkids $
CALC      ; loglu = logl $
PROBIT    ; Quiet
          ; Lhs = doctor ; Rhs = one,age,educ,public $
CALC      ; loglr = logl $
CALC      ; List ; lrtest = 2*(loglu - loglr) ; cvalue = Ctb(.95,2) $
```

```
[CALC] LRTEST  =      7.8459659
[CALC] CVALUE  =      5.9914645
Calculator: Computed  2 scalar results
```

R13.4.2 Imposing and Testing Restrictions

Section R13.6.1 describes a specification that can be used to impose fixed value and equality restrictions in any model. The preceding test could be carried out with this device. The two probit commands could be specified with

```
PROBIT    ; Lhs = doctor ; Rhs = one,age,educ,public,married,hhkids
          ; Rst = b1,b2,b3,b4,b5,b6 $
```

and

```
PROBIT    ; Lhs = doctor ; Rhs = one,age,educ,public,married,hhkids
          ; Rst = b1,b2,b3,b4,0,0 $
```

The second **PROBIT** command estimates the model subject to the restrictions that the last two coefficients equal zero. (The **; Rst = list** specification in the first model is actually redundant since it does not specify any restrictions. It simply names the parameters in the model.)

R13.4.3 Homogeneity of Models in a Stratified Data Set

The following example demonstrates several features of *LIMDEP* and shows a general method of testing whether the same model should be used for all subgroups in a sample. The modeling framework is

For group i, model $f_i(.)$ applies to observations y_{ig}, $\mathbf{x}_{ig}$, $g = 1,\ldots,G$ groups, $i = 1,\ldots,n_g$.

H_0: The same model applies to all groups.

H_1: The form of the model is the same for all groups, but the parameters differ across groups.

A likelihood ratio test of the null hypothesis is carried out as follows:

Unrestricted: $\log L_u = \sum_{g=1}^{G} \log L_g$; models are fit separately.

Restricted: $\log L_r$ = the log likelihood for the model with all observations pooled.

The chi squared test statistic has degrees of freedom $(G\text{-}1)\times K$ where K is the number of parameters in the model and is computed as

$$\chi^2 = 2(\log L_u - \log L_r).$$

This test is equivalent to a 'Chow test' in the linear regression model.

The following *LIMDEP* procedure does this computation. It assumes that the sample stratification is provided by a variable that is coded 1,2,…,G. (See Chapter R4 if you need to create this variable from some other kind of indicator variable.)

```
PROC = samemodl(Model, y, x, group)     $
CALC      ; g = Max(group)              ? How many groups ?
          ; loglu = 0                   $ Will be accumulated
DO FOR    ; 100 ; grp = 1,g             $ Execute once for each stratum
INCLUDE   ; New ; group = grp           $ Select the observations
Model     ; Lhs = y ; Rhs = x ; Quiet   $ Estimate the model
CALC      ; loglu = loglu + logl        $Unrestricted log likelihood = sum
ENDDO     ; 100                         $ End of repetition block
SAMPLE    ; All                         $ Full sample for restricted model
Model     ; Lhs = y ; Rhs = x ; Quiet   $ Estimate model using full sample
CALC      ; loglr = logl                ? Retrieve restricted log likelihood
          ; List ; lrtest = 2*(loglu - loglr) ? LR statistic
          ; df = (g-1)*kreg             ? Degrees of freedom
          ; prob = 1 - Chi(lrtest,df)   $ P value = significance level
ENDPROC $
```

This procedure can be used for any model that is built up simply for a dependent variable and a set of independent variables. If need be, you could modify the model command for some other modeling framework. Note that the model command is generic. You could use this for a probit model, then with exactly the same program change over to a logit model.

TIP: Note that the model command includes **; Quiet**. This will suppress what might be a huge amount of output. This will often be a good idea.

To illustrate, the following tests for homogeneity across genders of a probit model.

```
NAMELIST    ; x = one,age,married,hhkids $
CREATE      ; sex = female + 1 $
EXECUTE     ; proc = samemodl(probit,doctor,x,sex) $
```

```
[CALC] LRTEST  =    584.5646542
[CALC] DF      =      4.0000000
[CALC] PROB    =       .0000000
Calculator: Computed   4 scalar results
Maximum repetitions of PROC
```

All other output from the procedure has been suppressed, so it only reports the outcome of the test.

R13.4.4 Testing for Equal Coefficient Vectors

The test procedure in the previous section was constructed to illustrate using a procedure to carry out a repetitive operation – in the application, the same calculations were applied to several subsamples. The particular test carried out there, testing for equality of the coefficients across subgroups of the sample, is so common that we have automated the entire computation in a single command. The general syntax for the test is

```
Model       ; For [(test) group variable] ; the model specification $
```

The specification of the group variable can be a set of values, such as

```
            ; For [(test) firm = 2,3,4,5] ; …
```

However, we note a caution, if the list of values does not exhaust the full sample, then the test will not be carried out correctly because the null specification uses the entire sample, not the pooled sample from the values given. The way to set this up correctly would be to set the pooled sample at the outset. For this example, suppose the full sample were firms 1,…,8. Then you would want the pooled sample to include firms 2,3,4,5. The way to proceed would be

```
INCLUDE     ; New ; firm >= 2 & firm <= 5 $
Model       ; For [(test) firm = 2,3,4,5] ; … $
```

For the example below, we replicated the test in the previous section with

```
PROBIT      ; For [(test) female]
            ; Quiet ; Lhs = doctor ; Rhs = one,age,married,hhkids $
```

The results are

```
+-------------------------------------------------------+
| Setting up an iteration over the values of FEMALE    |
| The model command will be executed for   2 values    |
| of this variable.  In the current sample of   27326 |
| observations, the following counts were found:       |
| Subsample   Observations    Subsample  Observations |
| FEMALE   =   0      14243    FEMALE  =   1     13083 |
| FEMALE   =****      27326                             |
+-------------------------------------------------------+
| Actual subsamples may be smaller if missing values   |
| are being bypassed.  Subsamples with 0 observations |
| will be bypassed.                                     |
+-------------------------------------------------------+

*******************************************************************
*       Subsample analyzed for this command is FEMALE   =       0 *
*******************************************************************

*******************************************************************
*       Subsample analyzed for this command is FEMALE   =       1 *
*******************************************************************

*******************************************************************
*       Full pooled sample is used for this iteration.            *
*******************************************************************
---------------------------------------------------------------------
Homogeneity Test for Estimated Model
---------------------------------------------------------------------
The model was estimated for  2 subsamples and the full sample
The likelihood ratio statistic is 2[Sum(g=1...G)logL(g)  -logL(pooled)]
Chi squared =    584.5647   Estimated degrees of freedom =   4
Estimated P value for this test is  .0000
---------------------------------------------------------------------
```

Note that the built in function does not require the group variable to be coded 1,2,…,G. It only expects to find a set of integer values. Thus, in the procedure, we used sex = female+1 which is now coded 1,2 while in the built in function, we used female = 0,1.

R13.4.5 Two Part Models: Cragg's Model for a Censored Dependent Variable

The tobit model specifies that $y = \max(0,\boldsymbol{\beta}'\mathbf{x}+\varepsilon)$, $\varepsilon \sim N(0,\sigma^2)$. It follows that the appropriate model for $d = 0$ if $y = 0$, $d = 1$ otherwise is a probit model with parameters $\boldsymbol{\gamma} = \boldsymbol{\beta}/\sigma$. Cragg's specification allows the parameters in the implied probit equation to differ completely from those in the tobit model, so that the complete model is a probit model for d and a separate truncated regression model for the positive values of y. (See Section E45.9.2 for further discussion.) Since the tobit log likelihood is simply the sum of the probit and truncated regression log likelihoods (see Greene, 2012), a simple test of the tobit model as a restriction on Cragg's (1971) model ($\boldsymbol{\gamma} = \boldsymbol{\beta}/\sigma$) can be based on

$$\chi^2 = 2(\log L_{probit} + \log L_{truncated\ regression} - \log L_{tobit}).$$

This will have degrees of freedom equal to the number of variables in **x**. The following does the test:

```
NAMELIST   ; x = ... your definition $
TOBIT      ; Lhs = y ; Rhs = x $
CALC       ; ltobit  =  logl $
CREATE     ; d = y > 0 $
TRUNC      ; Lhs = y ; Rhs = x $  This skips points with y = 0.
CALC       ; ltrunc = logl $
PROBIT     ; Lhs = d ; Rhs = x $
CALC       ; lprobit  = logl ; List
           ; lr = 2 * (ltrunc + lprobit - ltobit)
           ; df = Col(x) ; prob = Chi(lr,df) $
```

R13.4.6 Likelihood Ratio Tests for Discrete Choice Models

In many discrete choice models (probit, logit, ordered probit, Weibull, etc.), the log likelihood function for a model with only a constant term is

$$\log L_0 \;=\; \sum\nolimits_{j=outcomes} n_j \log p_j$$

where p_j is the proportion of the sample observations which have dependent variable equal to choice j, n is the sample size, and $n_j = np_j$. In this case, *LIMDEP* will carry out a likelihood ratio test of the hypothesis that all model parameters except the constant term are zero and report the results with other model output. The results below show the leading model output for a probit model, then a logit model fit with the same variables.

```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable                 DOCTOR
Log likelihood function    -17715.32374
Restricted log likelihood  -18019.55173
Chi squared [   3 d.f.]       608.45598 ←
Significance level               .00000
McFadden Pseudo R-squared      .0168832
Estimation based on N =  27326, K =   4
Inf.Cr.AIC  =35438.647 AIC/N =    1.297
Hosmer-Lemeshow chi-squared = 164.02100
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
Binary Logit Model for Binary Choice
Dependent variable                 DOCTOR
Log likelihood function    -17717.48126
Restricted log likelihood  -18019.55173
Chi squared [   3 d.f.]       604.14094 ←
Significance level               .00000
McFadden Pseudo R-squared      .0167635
Estimation based on N =  27326, K =   4
Inf.Cr.AIC  =35442.963 AIC/N =    1.297
Hosmer-Lemeshow chi-squared = 168.11271
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
```

Note that the restricted log likelihoods for the two models are identical. This follows from the earlier results since the restricted log likelihood is the same function of the sample proportions for both models.

A similar convenience arises in the Poisson regression model, in which the restricted parameter vector in a model with only a constant term is [log $\bar{y}$,0,…], so the log likelihood function for a restricted model can be computed at this 'estimate.' The output for a Poisson model includes

```
-----------------------------------------------------------------------------
Poisson Regression
Dependent variable                 DOCVIS
Log likelihood function   -105449.32913
Restricted log likelihood -108662.13583
Chi squared [   3 d.f.]      6425.61341
Significance level               .00000
McFadden Pseudo R-squared      .0295669
Estimation based on N =  27326, K =   4
Inf.Cr.AIC  =********* AIC/N =    7.718
Chi- squared =265073.54284  RsqP= .0460
G  - squared =157860.07745  RsqD= .0391
Overdispersion tests: g=mu(i)  : 20.807
Overdispersion tests: g=mu(i)^2: 20.565
--------+--------------------------------------------------------------------
```

The model results show the results of the overall test of model significance.

TIP: If your model output for the discrete choice model does not contain the results for this test, it is probably because you neglected to include a constant term in your Rhs. If you do omit the constant term, it is possible for the log likelihood for your model to be less than that for the model with only a constant term. Moreover, even if it is not, the test given above will be misleading since the model being tested is not nested in the larger model. That is, the model with **; Rhs = x** cannot be compared to the model with **; Rhs = one**, unless *x* contains *one*.

R13.4.7 Likelihood Ratio Tests for Nonlinear Models

In most cases, the model with no coefficients is not a simple function of the sample data. In these cases, no simple test of overall significance is produced, but you can easily compute one. For example, for a tobit model, you can use

```
TOBIT     ; Lhs = y ; Rhs = one,… other variables $
CALC      ; loglu = logl ; kr = kreg $
TOBIT     ; Lhs = y ; Rhs = one $
CALC      ; loglr = logl
          ; lrtest  = 2*(loglu - loglr)
          ; df = kr - 1
          ; pvalue = 1 - Chi(lrtest,df) $
```

For this or any other case, you need only compare the log likelihood for your model with the log likelihood for a model which contains only a constant term.

The table below shows the output for the logit model of the previous section fit with only a constant term:

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
Dependent variable                DOCTOR
Log likelihood function    -18019.55173
Estimation based on N =  27326, K =   1
Inf.Cr.AIC  =36041.103 AIC/N =    1.319
Hosmer-Lemeshow chi-squared =  39.13032
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|     .52839***      .01252    42.19  .0000      .50385    .55294
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The log likelihood for the model is the one which appears in the earlier model results, so the test results will be the same.

R13.5 Lagrange Multiplier Tests

Lagrange multiplier (LM) tests are used similarly to likelihood ratio tests. These two, in contrast to the Wald test, rely specifically on the form of the likelihood function while the Wald statistic relies only on the large sample properties of the parameter estimator. The logic of the LM test is as follows: In estimating a model, if we do so without restrictions, the derivatives of the log likelihood will equal zero (to within rounding error) at the maximizer of the function. If we fit the model subject to restrictions, the maximized log likelihood will be lower (that is the basis of the LR test) and the derivatives of the full log likelihood function will not be zero. The test is based on measuring the extent to which the derivatives differ from zero. If the difference appears to be within the bounds of sampling variability, the hypothesis of the restrictions is not rejected.

The LM test has a significant shortcoming that weighs against a significant virtue. The test is the generally the most complicated of the three we are considering here, as it requires computation of the derivatives (and programming them). The appeal, however, is that the test is based entirely on the restricted model, which is often much simpler than the unrestricted model which is needed for the Wald and LR tests. For an example that we pursue below, the probit model with heteroscedasticity is a complicated model to estimate (and interpret). However, the test for heteroscedasticity can (using the LM statistic) be based entirely on a homoscedastic model, which is very simple.

LIMDEP has a built in feature that automates LM tests for most of the models supported by the program. The sections to follow describe this feature, then work through several examples..

R13.5.1 LM Tests Based on the Model Specification

For most of the nonlinear models, you can request *LIMDEP* to compute an LM statistic by treating your starting values as the restricted estimates. The procedure is then as follows:

1. Obtain the full *restricted* set of parameter estimates.
2. Use the following command:

Model Name ; ... usual setup ... ; Start = your values ; Maxit = 0 $

When you specify a model, provide starting values, then prevent iterations with **; Maxit = 0**, (you may specify **; LM test** instead of **; Maxit = 0**). *LIMDEP* does the following:

1. Computes the LM statistic using the starting values.
2. Reports the usual output, i.e., estimates, standard errors, etc. as if the starting values were the maximum likelihood estimates.
3. Reports the LM statistic with the final output.

TIP: To do a Lagrange multiplier test, therefore, you would obtain the starting values by estimating the restricted model, then specifying the unrestricted model as the command, providing as starting values the estimates from the restricted model. This usually includes some fixed values for parameters in the unrestricted specification, typically a set of zeros.

Example: Fixed Value Restriction in a Logit Model

The following commands fit a logit model with an interaction between age and gender. We are interested in testing the hypothesis that the coefficient on this term equals zero. The unrestricted model is

```
LOGIT        ; Lhs = doctor
             ; Rhs = one,age,educ,hhninc,married,female,age*female $
```

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
Dependent variable               DOCTOR
Log likelihood function    -17444.88483 ←
Restricted log likelihood  -18019.55173
Chi squared [   6 d.f.]      1149.33380
Significance level               .00000
McFadden Pseudo R-squared      .0318913
Estimation based on N =  27326, K =   7
Inf.Cr.AIC  =34903.770 AIC/N =    1.277
Hosmer-Lemeshow chi-squared =  80.07335
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient     Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|    -.59710***     .09733    -6.13  .0000     -.78786   -.40634
     AGE|     .02896***     .00159    18.23  .0000      .02585    .03208
    EDUC|    -.02697***     .00580    -4.65  .0000     -.03834   -.01560
  HHNINC|    -.18680**      .07571    -2.47  .0136     -.33518   -.03841
 MARRIED|    -.01076        .03134     -.34  .7313     -.07219    .05066
  FEMALE|    1.06350***     .10301    10.32  .0000      .86161   1.26539
        |Interaction AGE*FEMALE
Intrct01|    -.01141***     .00233    -4.89  .0000     -.01598   -.00684
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The large z statistic (-4.89) attached to the term in the unrestricted model suggests that the hypothesis should be rejected. We will now fit the model subject to the restriction. The obvious way to proceed is to drop the variable from the model, but for our purposes, it is more useful to compute the restricted estimator. The command, which uses the device shown in Section R13.6.1 is

```
LOGIT      ; Lhs = doctor
           ; Rhs = one,age,educ,hhninc,married,female,age*female
           ; Rst  = b1,b2,b3,b4,b5,b6,0$
```

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
Dependent variable                DOCTOR
Log likelihood function    -17456.85932 ←
Restricted log likelihood  -18019.55173
Chi squared [   6 d.f.]      1125.38482
Significance level               .00000
McFadden Pseudo R-squared      .0312268
Estimation based on N =  27326, K =   6
Inf.Cr.AIC  =34925.719 AIC/N =    1.278
Hosmer-Lemeshow chi-squared =  93.34713
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|    -.42017***      .09018    -4.66  .0000     -.59692   -.24342
     AGE|     .02382***      .00118    20.13  .0000      .02150    .02614
    EDUC|    -.02491***      .00578    -4.31  .0000     -.03624   -.01359
  HHNINC|    -.18201**       .07567    -2.41  .0162     -.33032   -.03371
 MARRIED|     .00785         .03106      .25  .8005     -.05303    .06873
  FEMALE|     .57669***      .02614    22.06  .0000      .52545    .62793
        |Interaction AGE*FEMALE
Intrct01|       .000    .....(Fixed Parameter).....
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
Fixed parameter ... is constrained to equal the value or
had a nonpositive st.error because of an earlier problem.
-----------------------------------------------------------------------------
```

The warning in the footnotes indicates that *LIMDEP* has noticed that a coefficient has a nonpositive estimated standard error. This usually results from what we have done, fixing a coefficient to zero. But, sometimes this signals a problem with the estimation of the model. On the basis of the two sets of results, we can compute a likelihood ratio test of the hypothesis. The statistic will be twice the difference in the log likelihoods, or 23.949. The square of the z statistic is 23.912, so thus far the results are consistent. We will now carry out the LM test, with

```
LOGIT      ; Lhs = doctor
           ; Rhs = one,age,educ,hhninc,married,female,age*female
           ; Start = b
           ; Maxit = 0 $
```

The starting value is the result of the previous estimation, which has a zero in the last position.

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
Dependent variable                DOCTOR
LM Stat. at start values        23.96151 ←
LM statistic kept as scalar     LMSTAT
Log likelihood function     -17456.85932
Restricted log likelihood   -18019.55173
Chi squared [   6 d.f.]       1125.38482
Significance level                .00000
McFadden Pseudo R-squared       .0312268
Estimation based on N =  27326, K =   7
Inf.Cr.AIC  =34927.719 AIC/N =    1.278
Hosmer-Lemeshow chi-squared =  93.34713
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient      Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|    -.42017***      .09708    -4.33  .0000     -.61044    -.22990
     AGE|     .02382***      .00158    15.11  .0000      .02073     .02691
    EDUC|    -.02491***      .00579    -4.30  .0000     -.03627    -.01356
  HHNINC|    -.18201**       .07567    -2.41  .0162     -.33033    -.03370
 MARRIED|     .00785         .03129      .25  .8019     -.05348     .06918
  FEMALE|     .57669***      .10275     5.61  .0000      .37531     .77807
        |Interaction AGE*FEMALE
Intrct01|       .000         .00233      .00 1.0000       -.005       .005
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

NOTE: When the restrictions are imposed, the estimated asymptotic covariance matrix will account for those restrictions, as can be seen in the second set of results. But, when you use **; Maxit = 0**, there is no way to know what the restrictions are, so all parameters are treated as free. Thus, the standard errors are different – generally larger – as we see in the results immediately above. This effect is clearly visible in the results above in a comparison of the two reports of parameter estimates and standard errors.

R13.5.2 LM Test of Homoscedasticity in a Probit Model

The probit model suggested earlier is a natural candidate for the LM test. The test can be carried out as follows:

NAMELIST **; x = ... variables in the regression part**
; z = ... variables in the heteroscedasticity $ No constant in z.
CALC **; m = Col(z) $** Number of restrictions. Keep it generic.
PROBIT **; Lhs = y ; Rhs = x $** Restricted model.
PROBIT **; Lhs = y ; Rhs = x ; Rh2 = z**
; Het ? This is the unrestricted model using restricted coefficients.
; Start = b, m_0 ; Maxit = 0 $
CALC **; List ; lmstat**
; 1 - Chi(lmstat,l) $

Note that the restricted estimator in this case is the probit model under H_0: homoscedasticity, plus a column of zeros for $\boldsymbol{\gamma}$. For our health care data, an application of the procedure above produces the following results: (The initial model results and some statistics have been omitted.)

```
NAMELIST    ; x = one,age,educ,hhkids
            ; z = female,married,hhninc $
PROBIT      ; Lhs = doctor ; Rhs = x $
PROBIT      ; Lhs = doctor ; Rhs = x ; Het ; Hfn = z ; Maxit = 0 ; Start = b,0,0,0 $
```

```
Maximum of     0 iterations. Exit iterations with status=1.
Maxit = 0. Computing LM statistic at starting values.
No iterations computed and no parameter update done.

-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               DOCTOR
LM Stat. at start values      336.00066 ←
LM statistic kept as scalar    LMSTAT
Log likelihood function    -17676.31452
Restricted log likelihood  -18019.55173
Chi squared [   6 d.f.]       686.47442
Significance level               .00000
McFadden Pseudo R-squared      .0190480
Estimation based on N =  27326, K =   7
Inf.Cr.AIC  =35366.629 AIC/N =    1.294
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|     .15472***      .05643     2.74  .0061      .04412    .26531
     AGE|     .01337***      .00119    11.24  .0000      .01104    .01570
    EDUC|    -.03090***      .00391    -7.90  .0000     -.03856   -.02323
  HHKIDS|    -.12402***      .01797    -6.90  .0000     -.15923   -.08880
        |Variance function
  FEMALE|       .000         .04717      .00 1.0000       -.092      .092
 MARRIED|       .000         .05163      .00 1.0000       -.101      .101
  HHNINC|       .000         .12116      .00 1.0000       -.237      .237
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

WARNING: You must provide the correct starting values for this procedure. Sometimes, the parameters that are estimated by the nonlinear procedure are transformations of the original parameters. The specific descriptions of the models describe the parameters that are estimated in each case, and indicate exactly the way to provide starting values. For two examples:

- **Tobit:** The model is parameterized in terms of $\boldsymbol{\beta}$ and σ. These are the values reported by the program in the output. But, the parameters estimated internally are $\boldsymbol{\gamma} = \boldsymbol{\beta}/\sigma$ and $\theta = 1/\sigma$. These are transformed to produce the output. LM statistics are based on these transformed parameters.
- **Parametric Survival Models:** The Weibull, log logistic, etc. models are reported as parameterized in terms of $\boldsymbol{\beta}$ and σ. But, the internal variance parameter is $P = 1/\sigma$.

As noted, the appropriate starting values to provide are given with the model description.

R13.5.3 LM Tests for the Linear Regression Model

There are a number of cases in which the LM statistic can be computed as a simple function of the R^2 in a linear regression. We consider several examples:

Breusch and Pagan's (1979) Test for Heteroscedasticity in the Classical Normal Regression Model

For testing H_0: Var[ε] = σ^2 against H_1: Var[ε] = f($\boldsymbol{\gamma}'\mathbf{z}$), in the classical regression $y_i = \boldsymbol{\beta}'\mathbf{x}_i + \varepsilon_I$ the Lagrange multiplier statistic is one half the explained sum of squares in the regression of e_i /($\mathbf{e}'\mathbf{e}/n$) on **z**. The residuals are computed from the homoscedastic regression. The test is applicable to linear or nonlinear regression.

Omitted Variables in Linear Regression

```
NAMELIST    ; x = ... ; z = ... $
REGRESS     ; Lhs = y  ; Rhs = x  ; Res = e $
CREATE      ; e2 = e^2 / (sumsqdev / n) $
CALC        ; lmstat = .5 * Xss(z,e2) $
```

The matrix algebra program provides a straightforward method of doing this computation. (See Greene (2012).)

```
NAMELIST    ; x = ... ; z = one,... $
REGRESS     ; Lhs = y ; Rhs = x ; Res = e $
CREATE      ; gi =  e*e / (sumsqdev / n)  -  1 $
MATRIX      ; lmstat = .5 * gi’z * <z’z> * z’gi $
```

Nonlinear Regression

This would be the same as above except that the **REGRESS** command is replaced with

```
NLSQ        ; Lhs = y
            ; Fcn = the model
            ; Start = ... ; Labels = ...
            ; Res = e $
```

Godfrey's (1978) LM Test for Autocorrelation

For testing for *P*th order moving average or autoregression in the disturbance of a classical regression, the LM statistic equals nR^2 in the regression of e_t on $e_{t-1},...,e_{t-P}$ and **X**, where e_t is an OLS residual. Missing values at the beginning of the series are filled with zeros.

```
NAMELIST    ; x = ... $
CREATE      ; e1 = 0 ; e2 = 0 ; ... ; ep = 0   $   (Do this for p variables)
REGRESS     ; Lhs = y ; Rhs = x  ; Res = e $
CREATE      ; If (_obsno > p) | e1 = e[-1]  ; e2 = e[-2] ; ...  ; ep = E[-p] $
CALC        ; lmstat = n * Rsq (x,e1,e2,...,ep,e) $
```

In this case, we create and fill with zeros the variables *e*1, *e*2,... before the second step estimator. It would have been possible simply to regress *e* on *x*,*e*[-1],... But, if we did so, the missing lagged values would be filled with -999s (the missing value code), not zeros as required.

R13.5.4 Programming Lagrange Multiplier Tests

Lagrange multiplier (LM) tests can be carried out by several methods. The matrix algebra package is well suited for this computation. In addition, several types of LM statistics can be computed for you by the estimation program for the model being analyzed. This section will describe and illustrate several approaches to computing LM statistics.

The Lagrange multiplier statistic for a test of hypothesis H_0 is LM = $\mathbf{g}_0'[\mathbf{H}_0]^{-1}\mathbf{g}_0$, where $\mathbf{g}$ is the gradient of the log likelihood function and $\mathbf{H}$ is n times a consistent estimator of the expected value of the Hessian of the log likelihood. The subscript '0' indicates that these matrices are to be computed at the parameter estimates obtained under the restrictions of the null hypothesis, H_0.

Example: Testing Homoscedasticity in a Probit Model Using Matrix Algebra

The log likelihood function for a probit model with multiplicative heteroscedasticity is

$$\log L = \Sigma_i \ln\Phi[q_i\boldsymbol{\beta}'\mathbf{x}_i \times \exp(-\boldsymbol{\gamma}'\mathbf{z}_i)]$$

where $q_i = 2y_i - 1 = \text{sgn}(y_i)$.

The gradient is $\partial\text{logL}/\partial\boldsymbol{\beta} = \Sigma_i v_i\mathbf{x}_i$

$$\partial\text{logL}/\partial\boldsymbol{\gamma} = \Sigma_i v_i\mathbf{z}_i(-\boldsymbol{\beta}'\mathbf{x}_i)$$

where $v_i = q_i(\phi_i/\Phi_i)\times\exp(-\boldsymbol{\gamma}'\mathbf{z}_i)$

and ϕ_i, Φ_i = standard normal PDF and CDF at $q_i\boldsymbol{\beta}'\mathbf{x}_i\times\exp(-\boldsymbol{\gamma}'\mathbf{z}_i)$.

The hypothesis to be tested is H_0: $\boldsymbol{\gamma} = \mathbf{0}$. For convenience, combine the two parts of the gradient into $\mathbf{g}_i$. The most convenient estimator to use for the LM test is usually the BHHH estimator of $\mathbf{H}$,

$$\mathbf{H} = \sum_{i=1}^{n}\mathbf{g}_i\mathbf{g}_i' = \sum_{i=1}^{n} v_i^2\begin{bmatrix}\mathbf{x}_i\mathbf{x}_i' & -\mathbf{x}_i\mathbf{z}_i'(\boldsymbol{\beta}'\mathbf{x}_i) \\ -\mathbf{z}_i\mathbf{x}_i'(\boldsymbol{\beta}'\mathbf{x}_i) & \mathbf{x}_i\mathbf{x}_i'(\boldsymbol{\beta}'\mathbf{x}_i)^2\end{bmatrix}$$

With $\boldsymbol{\gamma} = \mathbf{0}$, the LM statistic is simple to compute. The matrix function Bhhh is written specifically for this type of calculation. The function Lmm in **CREATE** simplifies calculation of the first derivative.

```
NAMELIST  ; x = ... ; z = ...          $ Note, z must not contain one!
CREATE    ; y = the dependent variable $
PROBIT    ; Lhs = y ; Rhs = x          $ Restricted model
CREATE    ; qi  = 2 * y - 1            ? -1 for y = 0, +1 for y = 1
          ; xb  = x'b                  ? beta'x
          ; gi  = -qi * Lmm(qi * xb)   ?qi * N01 / Phi
          ; vb  = gi                   ? d./db'x
          ; vg  = -gi * xb             $ d./dg'z
MATRIX    ; gb  = x'vb                 ? gradient for beta
          ; gg  = z'vg                 ? gradient for gamma
          ; g0  = [gb / gg]            ? stack the two vectors
          ; h0  = Bhhh(x,z,vb,vg)      ? BHHH form of the Hessian
          ; List
          ; lm = g0' <h0> g0           $ LM statistic reported
```

(A much easier approach for this application is shown in the next section.)

Example: LM Test for Groupwise Heteroscedasticity in Regression

Some particular Lagrange multiplier statistics have been derived explicitly and have a relatively simple form. The groupwise heteroscedastic regression (Greene (2012)) is an example:

$$\mathbf{y}_i = \mathbf{X}_i\boldsymbol{\beta} + \boldsymbol{\varepsilon}_i,\ T \text{ observations, } i = 1,...,G \text{ groups.}$$

$$H_0: \boldsymbol{\varepsilon}_i \sim \text{N}(\mathbf{0},\sigma^2\mathbf{I}).$$

The alternative hypothesis is that σ^2 differs by group, though $\boldsymbol{\beta}$ remains the same for all i. The LM statistic is

$$LM = (T/2)\Sigma_i(s_i^2/s^2 - 1)^2,$$

where s_i^2 is the group specific least squares residual variance and s^2 is the counterpart when the data are pooled. Variances are computed using T and GT as divisors, with no degrees of freedom corrections. The following procedure would compute the test statistic. In this example, we use **CALCULATE** instead of **MATRIX** in computing the test statistic. In the procedure below, the symbol '*g*' is the number of groups, which you would provide specifically. As usual, '*y*' is the dependent variable. Within the loop, we reset the sample after each regression.

```
NAMELIST    ; x = ... $
CREATE      ; y = your dependent variable $
SAMPLE      ; All observations $ The pooled sample has n = tg observations.
CALC        ; g = the appropriate value for the number of groups $
CALC        ; t = n / g  ; first = 1 ; last = t $
CALC        ; s2 = Ess(x,y) / n ;  lmstat = 0 $
PROCEDURE  $
SAMPLE      ; first - last $
CALC        ; s2j = Ess(x,y) / t
            ; lmstat = lmstat + (t / 2) * (s2j / s2  -  1)^2
            ; first = first + t ; last = last + t $
ENDPROCEDURE $
EXECUTE     ; j = 1,g  $  This says execute the procedure for j = 1 to g.
CALC        ; List ; lmstat $
```

The procedure above is a bit more convenient with a fixed subsample size than it would be otherwise, but not much. To modify it to account for variable sample sizes, we will require a stratification variable which allows us to partition the sample. Suppose that variable is named *group* and it takes values 1,2,…,*G*. The calculation of *G* for the program's purpose is

```
CALC         ; g = Max(group) $
```

Then, the procedure would be replaced with

```
PROCEDURE  $
INCLUDE     ; New ; group = j  $
CALC        ; s2j = Ess(x,y) / n
            ; lmstat = lmstat + (n / 2) * (s2i / s2  -  1)^2 $
ENDPROCEDURE $
```

R13.6 Estimation Subject to Restrictions

Section R13.2 showed how to impose linear restrictions in least squares regression. You can also impose restrictions on other models. Two procedures are provided. The first allows you to impose fixed value and equality restrictions on any estimated parameter vector in any model. Most applications that involve restrictions on parameters will be covered by this case. The second case is more general linear restrictions, which can also be imposed in most models.

Before describing these procedures, we note two important general cases, by way of practical suggestions.

Nonlinear Restrictions

Estimation subject to nonlinear restrictions raises a set of practical issues not present with linear restrictions. As a general rule, nonlinear restrictions, such as

$$\beta_1^2 + \beta_2^2 + \beta_3^2 = c^2$$

which restricts the three parameters to lie on the surface of a ball with radius c, requires more elaborate tools than are used in the general model estimation programs in *LIMDEP*. For specific cases, you might be able to program the restrictions directly into your own likelihood function in **MAXIMIZE**.

Inequality Restrictions

LIMDEP does contain an estimator for linear regression with inequality restrictions, using linear and quadratic programming methods. These are described in the *Econometric Modeling Guide*. More generally, however, we will not make use of general inequality restrictions such as

$$\mathbf{R\beta} - \mathbf{q} >> \mathbf{0}.$$

There are a few common cases that do appear regularly. Two in particular are models that contain a variance parameter, σ, that must be forced to be positive and models that contain a correlation coefficient, ρ, that must lie in (-1,1). The typical way to handle the first of these is to reparameterize the model in terms of $\sigma = \exp(\theta)$, and estimate θ which is unrestricted. For the correlation coefficient, the standard approach, and the one used here, is to formulate the model in terms of the hyperbolic arctangent function, $\theta = 1/2 \ln[(1+\rho)/(1-\rho)]$. The structural parameter, θ, is unrestricted, and $\rho = [\exp(2\theta)-1]/[\exp(2\theta)+1]$, which is bounded in the interval. We find in general, however, that in cases in which this device is employed, the unrestricted estimators of σ and ρ obey the restriction anyway.

In practical terms, there is an element of this aspect of estimation that the user should be mindful of. Restricting a parameter such as σ and ρ as suggested above does not generally force the optimizer to find an interior solution that it would not have found otherwise. That is why the restriction/retransformation is not actually necessary in most cases. When the restriction tends to be binding, as sometimes happens with the bivariate probit model, for example, what you will find is that θ will be drifting off to $+\infty$ or $-\infty$, so that ρ will be getting close to -1 or +1. The force of the restriction is to prevent the program from dividing by zero or taking the log of a negative number. It does not make the solver find a better solution for the parameter.

R13.6.1 Fixed Value and Equality Restrictions

All models in *LIMDEP* can be estimated subject to equality and/or fixed value restrictions on the parameters. These can be cross equation restrictions, such as in the multinomial logit model or switching regression models, in which you might want to force one coefficient vector to equal another, or within equation restrictions, such as in any regression model in which you want to force coefficients to equal each other or fixed values (or both).

The command structure used to request this feature is

```
Model           ; ... ; Rst = the specification
```

Restrictions are specified by just giving a list of labels for the parameters. Repetitions of labels imply equality restrictions. Instead of a label, you may give a fixed value. The parameter which is fixed is not reestimated; it is forced to the value you provide.

For example, suppose the choice variable in a logit model, *y*, is explained by a constant, *educ*, and *income*. Assume *y* takes four values. It is desired to force the income coefficient to be the same in all three parameter sets. The model contains nine parameters. The command could be

```
LOGIT           ; Lhs = y; Rhs = one,income,educ
                ; Rst = b1,b2,b3,b4,b2,b5,b6,b2,b7 $
```

Note that two constraints are imposed. If it were desired to force the last coefficient to equal .1, say, it would be necessary only to change *b*7 to .1. Note, as well, there is nothing implied by the consecutive numbers used for the parameters. The sequence of symbols

```
                ; Rst = aa,inc,ab,ca,inc,qr,ty,inc,dc
```

would have exactly the same effect. The crucial element is that the second, fifth and eighth symbols are the same. The other symbols in the list can be anything, as long as they are different. That said, when we use this feature, we usually choose convenient combinations of numbers and letters that make the specification easy to understand. The list of symbols that you choose in **; Rst** are purely for internal use by the optimizer. They will not appear anywhere in your results for the model.

TIP: Forcing a coefficient to equal a fixed value in a logit model is not the same as forcing the corresponding marginal effect to equal a fixed value. Aside from zero, fixed values in the logit model are going to be difficult to interpret. Of course, using fixed value constraints does provide an easy way to test hypotheses.

In order to use this feature, you will need to know what the precise parameter layout is for the model you are estimating. This will be given with the specific model descriptions in the chapters to follow. One way to find out in many cases is to fit the model without restrictions. If you provide the wrong specification for the Rst list, you will get a diagnostic error about syntax in the restrictions. Here is an example, based on a tobit model:

```
SAMPLE          ; 1-1000 $
CREATE          ; x1 = Rnn(0,1) ; x2 = Rnn(0,1) ; y = 0!(x1 + x2 + Rnn(0,2)) $
CALC            ; Rng(1) $
CALC            ; Ran(12345) $
TOBIT           ; Lhs = y ; Rhs = one,x1,x2 ; Rst = b0,b1,b1 $

Expected   4 specifications in RST/CML list. Found   3.
```

This model command specifies a tobit model with two independent variables, and attempts to force the two coefficients to be equal. The problem with this command is that the tobit model has an additional parameter, the standard deviation, σ. Since the restriction list contains no specification for σ, a syntax error is indicated.

If the preceding example is respecified correctly, the following output results:

TOBIT ; Lhs = y ; Rhs = one,x1,x2 ; Rst = b0,b1,b1,v $

```
Normal exit:   5 iterations. Status=0, F=    1339.909
-----------------------------------------------------------------------------
Limited Dependent Variable Model - CENSORED
Dependent variable                      Y
Log likelihood function     -1339.90941
Estimation based on N =   1000, K =   3
Inf.Cr.AIC  = 2685.819 AIC/N =    2.686
Threshold values for the model:
Lower=     .0000     Upper=+infinity
LM test [df] for tobit=     14.513[  3]
Normality Test, LM    =      1.890[  2]
ANOVA  based fit measure =    .149337
DECOMP based fit measure =    .295321
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Primary Index Equation for Model
Constant|     .06266         .08224      .76  .4461     -.09854    .22385
      X1|     .98991***      .05726    17.29  .0000      .87769   1.10213
      X2|     .98991***      .05726    17.29  .0000      .87769   1.10213
        |Disturbance standard deviation
   Sigma|    2.00750***      .06927    28.98  .0000     1.87174   2.14326
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

There are some shortcuts available for the **; Rst** specification:

- k_value means repeat the value k times. Thus, 3 _ .2 = .2, .2, .2.
- k_label means label1, label2,..., labelk. Thus, 3_beta = beta1, beta2, beta3.

These can be used to impose multiple restrictions. Thus,

3_beta, 3_beta = beta1, beta2, beta3, beta1, beta2, beta3
3_beta, 2_0, beta1 = beta1, beta2, beta3, 0, 0, beta1

If you provide starting values for the iterations for your model, you can use some or all of them as fixed values in the model. The symbol for a starting value is ( ) with nothing in the parentheses. A simple ( ) or k_( ) appears in your list at the corresponding point where the fixed value would appear. Thus,

; Start = .1, .2, .3, .4, 0, -.1, 0, 0,
; Rst = b1, ( ), b3, b4, b5, 3_( )

fixes the second and sixth to eighth parameters to the values in the starting values list.

NOTE: The labels you use in this specification are temporary, and will not appear anywhere in your output. They are used only for the purpose of specifying the restrictions in the command.

To illustrate this feature, we will fit a Heckman and Singer (1983) specification for a three class latent class model. The Heckman and Singer form of latent class model is one in which the parameters for all classes are the same except for the constant term. In this form, we can think of the model as a random effects model in which the random component has a discrete distribution. The base model is a binary logit model. The following fits the model subject to all the restrictions implied by the Heckman and Singer latent class assumption.

```
LOGIT    ; Lhs = doctor ; Rhs = one,age,educ,married,hhninc,hsat
         ; Pds = ti
         ; Lcm ; Pts = 3
         ; Rst = a1, 5_beta, a2, 5_beta, a3, 5_beta, theta1, theta2, theta3 $
```

```
-----------------------------------------------------------------------------
Logit    Regression Start Values for DOCTOR
Dependent variable                DOCTOR
Log likelihood function    -16639.50194
Estimation based on N =  27326, K =   6
Inf.Cr.AIC  =33291.004 AIC/N =    1.218
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    2.28258***      .10496    21.75  .0000     2.07685   2.48831
     AGE|     .01356***      .00123    10.98  .0000      .01114    .01598
    EDUC|    -.02576***      .00588    -4.38  .0000     -.03728   -.01423
 MARRIED|     .01395         .03187      .44  .6617     -.04852    .07641
  HHNINC|    -.01849         .07801     -.24  .8127     -.17139    .13441
    HSAT|    -.29189***      .00681   -42.87  .0000     -.30524   -.27855
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
Latent Class / Panel Logit    Model
Dependent variable                DOCTOR
Log likelihood function    -15617.83305
Restricted log likelihood  -16639.50194
Chi squared [  15 d.f.]      2043.33778
Significance level               .00000
McFadden Pseudo R-squared      .0614002
Estimation based on N =  27326, K =  10
Inf.Cr.AIC  =31255.666 AIC/N =    1.144
Unbalanced panel has   7293 individuals
LOGIT (Logistic) probability model
Model fit with  3 latent classes.
```

```
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Model parameters for latent class 1
Constant|    3.65647***      .20371    17.95  .0000     3.25720   4.05574
     AGE|     .02231***      .00205    10.88  .0000      .01829    .02633
    EDUC|    -.02965***      .01015    -2.92  .0035     -.04955   -.00976
 MARRIED|    -.03666         .04992     -.73  .4627     -.13451    .06119
  HHNINC|     .11252         .11083     1.02  .3100     -.10471    .32975
    HSAT|    -.33354***      .00951   -35.07  .0000     -.35219   -.31490
        |Model parameters for latent class 2
Constant|    -.22121         .28992     -.76  .4455     -.78945    .34703
     AGE|     .02231***      .00205    10.88  .0000      .01829    .02633
    EDUC|    -.02965***      .01015    -2.92  .0035     -.04955   -.00976
 MARRIED|    -.03666         .04992     -.73  .4627     -.13451    .06119
  HHNINC|     .11252         .11083     1.02  .3100     -.10471    .32975
    HSAT|    -.33354***      .00951   -35.07  .0000     -.35219   -.31490
        |Model parameters for latent class 3
Constant|    1.79550***      .21675     8.28  .0000     1.37068   2.22031
     AGE|     .02231***      .00205    10.88  .0000      .01829    .02633
    EDUC|    -.02965***      .01015    -2.92  .0035     -.04955   -.00976
 MARRIED|    -.03666         .04992     -.73  .4627     -.13451    .06119
  HHNINC|     .11252         .11083     1.02  .3100     -.10471    .32975
    HSAT|    -.33354***      .00951   -35.07  .0000     -.35219   -.31490
        |Estimated prior probabilities for class membership
Class1Pr|     .43883***      .04092    10.72  .0000      .35863    .51903
Class2Pr|     .10707***      .02364     4.53  .0000      .06074    .15340
Class3Pr|     .45410***      .02793    16.26  .0000      .39936    .50885
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

You may use the names of scalars rather than literal numbers as fixed values. As such, you may also use this feature to loop over coefficients, search for a parameter value, or fit a model at many different values. For example:

```
PROCEDURE $
POISSON      ; Lhs = num ; Rhs = x ; Rst = 8_b, beta9 $
ENDPROCEDURE $
EXECUTE      ; beta9 = .5 (.1) 1.5 $
```

estimates the model of our example with the coefficient on *logmth* taking fixed values of 0.5, 0.6, ..., 1.5.

You may also use the name of a scalar rather than a fixed number when you specify the number of values. This is useful in models in which you scan over the values of a parameter. For example, the BURR model is a logit model that contains an asymmetry parameter. The following would scan over a set of values of the extra parameter.

```
PROCEDURE $
BURR         ; Lhs = doctor ; Rhs = one,age,educ,married,hhninc,hsat
             ; Rst = 6_beta, lambda $
ENDPROCEDURE $
EXECUTE      ; lambda = .4(.1).7 $
```

NOTE: In *LIMDEP* Version 9, the list of values provided for this kind of looping procedure was assumed to be in the form **; name = first,last,step**. The syntax used above is **; name = first(step)last**. You may use either syntax in your **EXECUTE** command.

Finally, since the value used in Rst may be a variable scalar, you can use this to change the size of the model. The following general setup uses that feature

```
NAMELIST    ; x = a list of variables $
NAMELIST    ; z = another list of variables $
CALC        ; k = Col(x) ; m = Col(z) $
Model       ; Lhs = … ; Rhs = x ; Rst = k_beta, m_0 $
```

The number of parameters in the model is $k+m$. The specified restriction allows k free parameters related to x but forces the m parameters related to z to be zero.

R13.6.2 General Linear Restrictions

All nonlinear models estimated by maximum likelihood may be fit subject to linear equality restrictions on the parameters. The syntax is the same as that for restrictions on the linear least squares estimator described earlier. That is:

```
Model       ; … other setup
            ; CML: linear restriction, linear restriction, … $
```

Linear restrictions are specified as

$a1\ b(.) \pm a2\ b(.) \ldots = q1$, … as many restrictions as desired.

The coefficients $a1$, $a2$, … are specific values. If any are equal to 1.0, they may be omitted. An example appears below. Model coefficients are indexed by their appearance in the model specification and are indexed sequentially, almost always corresponding to a list of right hand side variables. We will examine two examples below. The restrictions may not be inequality restrictions. Thus, this feature can be used to force a set of coefficients to equal zero, but it cannot be used to make a sum of coefficients be greater than or equal to zero.

For an example, consider a Poisson regression model with conditional mean function,

$$\begin{aligned} logE[num] = b(1) &+ b(2) \times a + b(3) \times c + b(4) \times d + b(5) \times e + b(6) \times c67 + b(7) \times c72 \\ &+ b(8) \times c77 + b(9) \times logmth. \end{aligned}$$

The model is fit without restrictions using

```
NAMELIST    ; x = one,a,c,d,e,c67,c72,c77,logmth $
POISSON     ; Lhs = num ; Rhs = x $
```

We can use **; Rst = list** to constrain $b(9)$ to equal 1.0. The same restriction can be imposed with

```
POISSON     ; Lhs = num ; Rhs = x ; CML: b(9) = 1.0 $
```

The results produced by this will be identical Consider a more involved example. The four type dummy variables, *a*, *c*, *d*, and *e*, are included with an overall constant term; type *other* is dropped to avoid the multicollinearity problem of a complete set of dummy variables. Suppose, instead, we include all five groups, and constrain the coefficients to sum to zero. The constraint solves the identification problem.

```
CREATE      ; other = 1-a-c-d-e $
NAMELIST    ; x = one,a,c,d,e,other,c67,c72,c77,logmth $
POISSON     ; Lhs = num ; Rhs = x
            ; CML: b(10) = 1, b(2) + b(3) + b(4) + b(5) +b(6) = 0 $
```

HOW IT'S DONE: See Section R13.6.3 for technical details on linearly constrained maximum likelihood estimation.

NOTE: The specification **; CML:** cannot recognize the model specifications used by **; Test:** that are based on variable names rather than parameter numbers. **; CML** is meant to apply not only to regression style models, but also to settings such as **NLSQ** in which there is not a necessary natural association between parameters and variables.

Briefly, a final example is provided by the multinomial logit model. Consider a model with four outcomes and four attributes:

$$\text{Prob}[y_i = j] = e_{ij} /[e_{i0} + e_{i1} + e_{i2} + e_{i3}],\ j = 0,1,2,3;\ e_{ij} = \exp(\boldsymbol{\beta}_j'\mathbf{x}_i).$$

For identification, this model will be estimated subject to $\boldsymbol{\beta}_0 = \mathbf{0}$. Thus, with four attributes (including constant terms), $\boldsymbol{\beta}_1$ is $[b(1),b(2),b(3),b(4)]$, $\boldsymbol{\beta}_2 = [b(5),b(6),b(7),b(8)]$, and so on. Now, suppose for reasons unknown to us, you wished each element in $\boldsymbol{\beta}_2$ to equal twice its counterpart in $\boldsymbol{\beta}_1$. Your **LOGIT** command might appear as follows:

```
LOGIT       ; Lhs = yij
            ; Rhs = one,x1,x2,x3
            ; CML:  b(5)-2b(1) = 0, b(6)-2b(2) = 0, b(7)-2b(3) = 0, b(8)-2b(4) = 0 $
```

R13.6.3 Imposing Linear Constraints on Maximum Likelihood Estimators

The objective is to maximize log $L(\mathbf{data}, \boldsymbol{\theta})$ with respect to the parameter vector $\boldsymbol{\theta}$, subject to the set of linear constraints, $\mathbf{R}\boldsymbol{\theta} - \mathbf{q} = \mathbf{0}$, where

$\boldsymbol{\theta}$ = $K\times 1$ vector of constrained parameters,

$\mathbf{R}$ = $J\times K$ matrix of coefficients in J constraints,

$\mathbf{q}$ = $J\times 1$ vector of constants.

One approach to solving the maximization problem, which is equivalent to 'solving out the constraints,' is to partition $\mathbf{R}$ and $\boldsymbol{\theta}$ so that we may write them as

$$\mathbf{R}_1\boldsymbol{\theta}_1 + \mathbf{R}_2\boldsymbol{\theta}_2 = \mathbf{q}$$

such that $\mathbf{R}_1$ is J columns of $\mathbf{R}$, $\mathbf{R}_2$ is K-J columns, and $\mathbf{R}_1$ is nonsingular. We could then write

$$\boldsymbol{\theta}_1 = \mathbf{R}_1^{-1}(\mathbf{q} - \mathbf{R}_2\boldsymbol{\theta}_2).$$

One would then estimate $\boldsymbol{\theta}_2$ without constraints and solve for $\boldsymbol{\theta}_1$ residually. The method will be effective, but necessitates a possibly cumbersome search for the linearly independent columns of $\mathbf{R}$ and an inconvenient rearrangement of the elements of $\boldsymbol{\theta}$ to accommodate it.

The solution method used here, which is equivalent (it would always give the same answer) is to maximize the log likelihood function in terms of a (K-J) coefficient vector, $\boldsymbol{\gamma}$, such that $\boldsymbol{\gamma}$ is unconstrained, where $\boldsymbol{\theta} = \boldsymbol{f}(\gamma)$, and $\boldsymbol{\theta}$ satisfies the constraints. (The preceding is included in this general method.) Assuming that the constraints are linearly independent, there are K-J free, unconstrained parameters in $\boldsymbol{\gamma}$. To reparameterize the objective function in terms of this $\boldsymbol{\gamma}$, we begin with the spectral decomposition of

$$\mathbf{Q} = \mathbf{I} - \mathbf{R}'(\mathbf{R}\mathbf{R}')^{-1}\mathbf{R}.$$

Note that $\mathbf{Q}$ is a $K\times K$ idempotent matrix with rank K-J. Therefore, $\mathbf{Q}$ has K-J unit characteristic roots and J zero roots. Define the matrices $\boldsymbol{\Lambda}_1 = \mathbf{I}_{K-J}$ = the K-J unit characteristic roots of $\mathbf{Q}$, and $\boldsymbol{\Lambda}_2 = \mathbf{0}_J$ = the J zero characteristic roots. Define $\mathbf{C}_1$ = the $K\times$ (K-J) matrix whose columns are the characteristic vectors of $\mathbf{Q}$ corresponding to the unit roots and define $\mathbf{C}_2$ = the $K\times J$ matrix whose columns are the characteristic vectors of $\mathbf{Q}$ corresponding to the zero roots.

Let $\boldsymbol{\gamma}$ be the (K-J)$\times$1 vector of free parameters. Let

$$\mathbf{a} = \mathbf{C}_2(\mathbf{R}\mathbf{C}_2)^{-1}\mathbf{q}.$$

Then, the constrained parameter vector is

$$\boldsymbol{\theta} = \mathbf{C}_1\boldsymbol{\gamma} + \mathbf{a}.$$

It remains to show that $\boldsymbol{\theta}$ does satisfy the constraints $\mathbf{R}\boldsymbol{\theta} = \mathbf{q}$. By simple multiplication, it is obvious that $\mathbf{Ra} = \mathbf{q}$, so what remains to show is that $\mathbf{RC}_1 = \mathbf{0}$. By multiplication, $\mathbf{RQ} = \mathbf{0}$. But, by definition,

$$\mathbf{Q} = \mathbf{C}_1\boldsymbol{\Lambda}_1\mathbf{C}_1' + \mathbf{C}_2\boldsymbol{\Lambda}_2\mathbf{C}_2'.$$

Since $\boldsymbol{\Lambda}_2 = \mathbf{0}$, $\mathbf{RC}_1\boldsymbol{\Lambda}_1\mathbf{C}_1' = \mathbf{0}$. Post multiply by $\mathbf{C}_1$. Recall, $\boldsymbol{\Lambda}_1 = \mathbf{I}$ and, by construction of characteristic vectors, $\mathbf{C}_1'\mathbf{C}_1 = \mathbf{I}$. Of course, $\mathbf{0C}_1 = \mathbf{0}$, so we are left with $\mathbf{RC}_1\mathbf{I}\,\mathbf{I} = \mathbf{0}$, which completes the proof.

The estimation strategy, then, is to estimate $\boldsymbol{\gamma}$. We begin the iterations with any $\boldsymbol{\gamma}_0$. When $\boldsymbol{\theta}$ is to be used in any computation (function, derivatives), we compute $\boldsymbol{\theta} = \mathbf{C}_1\boldsymbol{\gamma} + \mathbf{a}$. The function and all derivatives are computed as functions of $\boldsymbol{\theta}$. Then,

$$\partial\text{objective}/\partial\boldsymbol{\gamma} = \mathbf{C}_1'\times\partial\text{objective}/\partial\boldsymbol{\theta}$$

$$\partial^2\text{objective}/\partial\boldsymbol{\gamma}\partial\boldsymbol{\gamma}' = \mathbf{C}_1'\times\partial^2\text{objective}/\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}'\times\mathbf{C}_1.$$

If needed, the reverse transformation from $\boldsymbol{\theta}$ to $\boldsymbol{\gamma}$ is $\boldsymbol{\gamma} = \mathbf{C}_1'\boldsymbol{\theta}$. (The equality follows from $\mathbf{C}_1'\mathbf{C}_1 = \mathbf{I}$ and $\mathbf{C}_1'\mathbf{C}_2 = \mathbf{0}$.)

Iterative estimation for the constrained procedures consists of the following steps:

Step 1. Obtain starting values for $\boldsymbol{\theta}$, either by the program or user supplied. Sometimes the starting vector may not satisfy the constraints. For the problems analyzed by *LIMDEP*, this will generally not be a problem.

Step 2. Collapse starting vector to obtain $\boldsymbol{\gamma}$.

Step 3. Iteration:

a. Enter iteration with $\boldsymbol{\gamma}$.
b. Expand $\boldsymbol{\gamma}$ to obtain $\boldsymbol{\theta}$. By construction, $\boldsymbol{\theta}$ satisfies the constraints.
c. Compute the function, gradient, and if needed, the Hessian in terms of $\boldsymbol{\theta}$.
d. Compute derivatives for $\boldsymbol{\gamma}$ as shown above.

Step 4. Test for convergence and either exit or update $\boldsymbol{\gamma}$ and return to Step 3.

Given the way the iterations are constructed, technical output produced during iterations will be for the unconstrained optimization problem. Thus, if your model command includes **; Output = 3**, the derivatives displayed in the output will be with respect to $\boldsymbol{\gamma}$, not $\boldsymbol{\theta}$. At exit, we recover $\boldsymbol{\theta} = \mathbf{C}_1\boldsymbol{\gamma} + \mathbf{a}$ and we estimate the asymptotic covariance matrix for the estimator with $\mathbf{C}_1\mathbf{V}\,\mathbf{C}_1'$ where $\mathbf{V}$ is the estimated asymptotic covariance for the estimator of $\boldsymbol{\gamma}$.

R13.6.4 Restricted Linear Regression with Multicollinearity

In a linear regression, if the **X** matrix is multicollinear, the ordinary least squares estimator cannot be computed. However, if restrictions are imposed on the coefficients, the restrictions may serve to increase the rank of the problem so that the coefficients can still be computed and the coefficient vector is identified. Consider the model with a complete set of dummy variables estimated in Section R13.6.2, where for now, we omit the first constraint:

```
CREATE     ; other = 1-a-c-d-e $l
NAMELIST   ; x = one,a,c,d,e,other,c67,c72,c77,logmth $
REGRESS    ; Lhs = num ; Rhs = x
           ; CLS:  b(2) + b(3) + b(4) + b(5) +b(6) = 0 $
```

The ten column data matrix for this model is collinear; *a+c+d+e+other = one*, so that as stated, the model could not be fit by linear least squares. However, the constraint in the last line turns this into a nine dimension problem, and makes it estimable. Nonetheless, the typical textbook approach to estimation in this case would break down, because the standard treatment in most textbooks and most software is to fit the unconstrained model first by least squares, then compute the restricted least squares estimator based on the unrestricted one. The textbook formula,

$$\mathbf{b}_c = \mathbf{b} - (\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'\,[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}(\mathbf{Rb} - \mathbf{q}),$$

requires inversion of $\mathbf{X}'\mathbf{X}$ prior to computation of the constrained estimator. But, since this matrix is singular, the computation stalls at this point. *In computing a linear regression, LIMDEP detects this condition and fits the restricted model directly*, without attempting to invert $\mathbf{X}'\mathbf{X}$. Technical details of this result appear in Greene and Seaks (1991) and in Section E8.3 on the linear regression model where the results are detailed in full.

The results for a model in which this condition emerges are shown below. No diagnostic about multicollinearity would appear in the results, as the more general estimator has been used at the outset.

CREATE ; health = hsat + 1 $ Now coded 1,2,…,11
CREATE ; Expand(health) $

```
HEALTH   was expanded as _HEALTH_.
Largest value =  11.  11 New variables were created.
Category
  1  New variable = HEALTH01    Frequency=     447
  2  New variable = HEALTH02    Frequency=     255
  3  New variable = HEALTH03    Frequency=     642
  4  New variable = HEALTH04    Frequency=    1173
  5  New variable = HEALTH05    Frequency=    1390
  6  New variable = HEALTH06    Frequency=    4233
  7  New variable = HEALTH07    Frequency=    2530
  8  New variable = HEALTH08    Frequency=    4231
  9  New variable = HEALTH09    Frequency=    6172
 10  New variable = HEALTH10    Frequency=    3061
 11  New variable = HEALTH11    Frequency=    3192
Note, this is a complete set of dummy variables.  If
you use this set in a regression, drop the constant.
```

The program output includes a warning that *_health_* is a complete set of dummy variables so it is necessary to drop one of them from a regression. This can be done at the outset by using **; Expand(health,0)** in the **CREATE** command, which would produce

```
HEALTH   was expanded as _HEALTH_.
Largest value =  11.   0 New variables were created.
Category
  1  New variable = HEALTH01    Frequency=     447
  2  New variable = HEALTH02    Frequency=     255
  3  New variable = HEALTH03    Frequency=     642
  4  New variable = HEALTH04    Frequency=    1173
  5  New variable = HEALTH05    Frequency=    1390
  6  New variable = HEALTH06    Frequency=    4233
  7  New variable = HEALTH07    Frequency=    2530
  8  New variable = HEALTH08    Frequency=    4231
  9  New variable = HEALTH09    Frequency=    6172
 10  New variable = HEALTH10    Frequency=    3061
Note, the last category was not expanded. You may use
this namelist as is in a regression with a constant.
```

However, to continue the example, we will proceed with the first form of the result. We then specify the regression,

REGRESS ; Lhs = hhninc ; Rhs = one,age,educ,married,hhkids,_health_
; CLS: b(16) = 0$

The unrestricted regression cannot be computed because the last 11 columns of the **X** matrix sum to the first column. However, the command does not generate a warning about collinearity.

```
-----------------------------------------------------------------------------
Restricted   least squares regression ............
LHS=HHNINC   Mean                   =         .35208
             Standard deviation     =         .17691
             No. of observations    =          27326  Degrees of freedom
Regression   Sum of Squares         =        95.6962          14
Residual     Sum of Squares         =        759.481       27311
Total        Sum of Squares         =        855.178       27325
             Standard error of e    =         .16676
Fit          R-squared              =         .11190  R-bar squared =   .11145
Model test   F[ 14, 27311]          =      245.80300  Prob F > F*   =   .00000
Diagnostic   Log likelihood         =    10180.04000  Akaike I.C.   = -3.58186
             Restricted (b=0)       =     8558.60603
             Chi squared [ 14]      =     3242.86795  Prob C2 > C2* =   .00000
Restrictions F[  1, 27310]          =         .00000  Prob F > F*   =  1.00000
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  HHNINC|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .06076***      .00776     7.83  .0000      .04554    .07598
     AGE|    -.00012         .00010    -1.13  .2578     -.00032    .00009
    EDUC|     .02067***      .00045    46.45  .0000      .01980    .02154
 MARRIED|     .08595***      .00260    33.08  .0000      .08086    .09104
  HHKIDS|    -.02028***      .00238    -8.54  .0000     -.02494   -.01562
HEALTH01|    -.03547***      .00847    -4.19  .0000     -.05207   -.01886
HEALTH02|    -.03606***      .01088    -3.31  .0009     -.05739   -.01473
HEALTH03|    -.02461***      .00726    -3.39  .0007     -.03883   -.01039
HEALTH04|    -.00741         .00573    -1.29  .1961     -.01865    .00382
HEALTH05|    -.00113         .00539     -.21  .8345     -.01169    .00944
HEALTH06|    -.00595         .00395    -1.51  .1321     -.01370    .00180
HEALTH07|     .01114**       .00447     2.49  .0127      .00238    .01991
HEALTH08|     .01186***      .00392     3.03  .0025      .00418    .01955
HEALTH09|     .01610***      .00364     4.42  .0000      .00897    .02324
HEALTH10|     .01498***      .00423     3.54  .0004      .00670    .02327
HEALTH11|       .000    .....(Fixed Parameter).....
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
Fixed parameter ... is constrained to equal the value or
had a nonpositive st.error because of an earlier problem.
-----------------------------------------------------------------------------
```

You might note, this configuration of the problem produces a signature – note the F statistic in the model results. It shows that the restriction is not binding, which will always be the case if the restriction is what secures identification of the model..

R14: Functions of Parameters

R14.1 Introduction

This chapter describes a post estimation procedure for analyzing nonlinear functions of parameters. The starting point is estimation of the parameters of a model, $\boldsymbol{\beta}$ and computation of an estimate of the asymptotic covariance matrix for that estimator, $\boldsymbol{\Sigma}$. Call these $\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\Sigma}}$. Post estimation, we will compute functions of $\hat{\boldsymbol{\beta}}$ such as ratios of elements, partial effects, and other functions that will be of the form

$$\hat{\boldsymbol{\gamma}} = \mathbf{c}\left(\hat{\boldsymbol{\beta}}\right) \quad \text{or} \quad \hat{\boldsymbol{\gamma}} = \mathbf{c}\left(\hat{\boldsymbol{\beta}}, \overline{\mathbf{z}}\right) \quad \text{or} \quad \overline{\hat{\boldsymbol{\gamma}}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{c}\left(\hat{\boldsymbol{\beta}}, \mathbf{z}_i\right).$$

The procedures described here will be used for three calculations:

- computing the functions
- estimating the variances and covariances of the computed functions
- testing the hypothesis that $\boldsymbol{\gamma}(\boldsymbol{\beta}) = \mathbf{0}$ using the sample statistics.

The command used for all three computations is the **WALD** command.

R14.2 Covariance Matrices for Nonlinear Functions

Two methods are used to obtain the estimated covariance matrix for the set of functions. The delta method estimates the covariance matrix by computing the covariance matrix of a linear approximation to the set of functions. The method of Krinsky and Robb uses information about the asymptotic distribution of the estimator of $\boldsymbol{\beta}$. The logic of Krinsky and Robb is to estimate the variance of a function of $\hat{\boldsymbol{\beta}}$ by sampling random draws from the asymptotic distribution of $\hat{\boldsymbol{\beta}}$ and obtaining an empirical estimate of the variance of the functions.

R14.2.1 The Delta Method

Suppose that **b** is a vector of parameter estimates of a parameter vector $\boldsymbol{\beta}$, computed by *any* procedure (or even, if you wished, by some other program). Suppose, as well, that $\boldsymbol{\Sigma}$ is the asymptotic covariance matrix of **b** and that **VARB** is our sample estimate of $\boldsymbol{\Sigma}$. Let $\gamma_1(\boldsymbol{\beta})$, $\gamma_2(\boldsymbol{\beta})$,...$\gamma_J(\boldsymbol{\beta})$ be J, up to 50, nonlinear functions of the form $\gamma_j(\boldsymbol{\beta})$. Let the vector $\boldsymbol{\gamma}(\boldsymbol{\beta})$ be the set of functions. Let $\mathbf{c}(\mathbf{b})$ denote the sample estimate of $\boldsymbol{\gamma}(\boldsymbol{\beta})$, obtained by computing $\boldsymbol{\gamma}(\boldsymbol{\beta})$ with the sample estimate, **b** and one of the three forms noted in Section R14.1. Denote by $\boldsymbol{\delta}_j$ the set of partial derivatives,

$$\partial\gamma_j(\boldsymbol{\beta})/\partial\boldsymbol{\beta}' = \boldsymbol{\delta}_j.$$

Note that $\boldsymbol{\delta}_j$ is a row vector. We estimate $\boldsymbol{\delta}_j$ with $\mathbf{d}_j$ by inserting our parameter estimates, **b** into the function defined by $\boldsymbol{\delta}_j$. Under the usual assumptions about well behaved estimates, the asymptotic covariance matrix for $\mathbf{c}(\mathbf{b})$ will be

$$\boldsymbol{\Gamma} = \boldsymbol{\Delta\Sigma\Delta}'$$

where the *j*th row of Δ is δ_j. The sample estimate of Δ is **D**, whose rows are $\mathbf{d}_j$. Our estimate of $\boldsymbol{\Gamma}$ is

$$\mathbf{G} = \mathbf{D} \times \mathbf{VARB} \times \mathbf{D}'.$$

For later purposes, we will call the matrix of derivatives, **D**, the Jacobian, though formally, the term would apply if $\boldsymbol{\gamma}(\boldsymbol{\beta})$ were itself a vector of derivatives, such as partial effects, so that Δ would then be a matrix of second cross partial derivatives. Here, $\boldsymbol{\gamma}$ need not be a vector of partial derivatives of a conditional mean function; $\boldsymbol{\gamma}(\boldsymbol{\beta})$ can be any set of functions you wish to analyze. The only requirements for the theory of the delta method to work are that $\boldsymbol{\gamma}(\boldsymbol{\beta})$ be continuous and continuously differentiable functions that do not involve the sample size.

When they involve the sample data, the functions will be of two general forms

$$\hat{\gamma} = \mathbf{c}\left(\hat{\boldsymbol{\beta}}, \overline{\mathbf{z}}\right) \quad \text{or} \quad \overline{\hat{\gamma}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{c}\left(\hat{\boldsymbol{\beta}}, \mathbf{z}_i\right).$$

In the first case, the functions are evaluated at the means of the data. In the second, the functions, themselves, are averaged over the sample observations. The third case noted in the introduction is $\hat{\gamma} = \mathbf{c}\left(\hat{\boldsymbol{\beta}}\right)$ in which the estimator is not a function of the sample data. For the first and third cases, the Jacobian is computed as

$$\hat{\Delta}\left(\hat{\boldsymbol{\beta}}, \overline{\mathbf{z}}\right) = \frac{\partial \mathbf{c}\left(\hat{\boldsymbol{\beta}}, \overline{\mathbf{z}}\right)}{\partial \hat{\boldsymbol{\beta}}'}.$$

That is, the Jacobian, like the functions, is computed at the sample means of the data. For the second case, the appropriate Jacobian (fortunately), is simply

$$\overline{\hat{\Delta}}\left(\hat{\boldsymbol{\beta}}\right) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathbf{c}\left(\hat{\boldsymbol{\beta}}, \mathbf{z}_i\right)}{\partial \hat{\boldsymbol{\beta}}'}.$$

The remaining part of the theory is the asserted asymptotic normal distribution of $\hat{\gamma}$ with asymptotic covariance matrix estimated by **G**.

R14.2.2 The Method of Krinsky and Robb

The method of Krinsky and Robb (1986) departs from **b** as an estimator of $\boldsymbol{\beta}$ and **V** = **VARB** as an estimator of $\boldsymbol{\Sigma}$, the covariance matrix of **b**. The covariance matrix for the estimator of $\hat{\gamma} = \mathbf{c}\left(\hat{\boldsymbol{\beta}}, \overline{\mathbf{z}}\right)$ is obtained by computing the empirical variance of R observations on $\hat{\gamma}$. We obtain R draws from the distribution of **b** then compute R draws from **c**(**b**). The draws on **b** are obtained using primitive draws from the multivariate standard normal distribution as follows. Let **L** be the Cholesky factorization of **V**, such that **LL′** = **V**. Then, a draw from the population of **b** is obtained as

$$\mathbf{b}_r = \mathbf{b} + \mathbf{L}\mathbf{v}_r.$$

Thus, $\mathbf{b}_r$ is a draw from the population which has mean $\mathbf{b}+\mathbf{L}\times\mathbf{0} = \mathbf{b}$ and variance $\mathbf{LIL}' = \mathbf{V}$. The draw on **b** is then transformed to a draw from **c** by computing $\mathbf{c}_r = \mathbf{c}(\mathbf{b}_r)$. The empirical variance is then estimated using

$$\mathbf{G} = \frac{1}{R}\sum_{r=1}^{R}\left(\mathbf{c}(\mathbf{b}_r) - \overline{\mathbf{c}}\right)\left(\mathbf{c}(\mathbf{b}_r) - \overline{\mathbf{c}}\right)'.$$

R14.3 The Wald Statistic

WALD is a general command for analyzing linear or nonlinear functions of parameters. The Wald statistic for testing the hypothesis $\boldsymbol{\gamma}(\boldsymbol{\beta}) = \mathbf{0}$ is computed as

$$W = \mathbf{c}'\mathbf{G}^{-1}\mathbf{c}.$$

This statistic has a limiting chi squared distribution with *J* degrees of freedom. As part of the results, **WALD** reports the computed value of *W*.

R14.4 The WALD Command

The general command for requesting a Wald statistic is

```
WALD      ; Labels  = a list of labels for the parameters
          ; Start   = the set of values for the parameters
          ; Var     = the asymptotic covariance matrix
          ; Fn1     = the first function
          ; Fn2     = the second function
          …         = ... up to 50 functions
          ; Keep    = list of functions  $
```

Request the Krinsky and Robb approach with the following addition to the **WALD** command:

; K&R ; Pts = number of draws.

The **; Pts** specification is optional. The default is 1,000 draws. **WALD** will compute functions that involve sample means of the data at the sample means. If the function you are computing is a sample mean of observations, such as an average partial effect, then add

; Average

to the command. The estimated variance is then computed appropriately.

NOTE: You should not use Krinsky and Robb with **; Average**, though the program will not stop you from doing so. *LIMDEP* will attempt to apply Krinsky and Robb to each term in the sum, which could lead to a huge amount of computation. However, this does not produce the correct covariance matrix, because the draws are treated as if they are independent when they are not – they use the same parameter vector. If you are computing a function with **; Average**, you should use the delta method.

TIP: There is no theoretical reason to prefer Krinsky and Robb's method to the delta method. Their 1986 paper that claimed otherwise was retracted in their 1990 paper that attributed the earlier finding to a software bug.

R14.4.1 Components of the WALD Command

The syntax of the command is such that the first three specifications provide the names, values of, and covariance matrix for a set of parameters, then one or more functions to be analyzed.

- The **; Labels = ...** specification is optional. Although they are optional, there is much less chance for confusion if you provide your own labels. If you do not provide the labels, the parameters will be labeled *b*1, *b*2, ..., *bK*, where *K* is the number of values you provide in the **; Start = ...** specification. Do note that when you define the functions, **; Fn1 = ...**, if you have not provided labels, you must use the *b*1,... given above.

- **; Start = ...** gives the numeric values for the parameters to be used in computing the functions. These may be given numerically, as in **; Start = 1.3, -.70248, .1114, 4** or they may be given symbolically, by using existing scalars and/or estimates from a previous model. For example, **; Start = b(1), b(2), rho, b(15), ssqrd**. You may use **; Parameters = the values** as a synonym

- **; Var = ...** specifies the covariance matrix to be used. The matrix must match the starting values in its dimensions. You may provide it three ways:

    - name of a matrix, as in **; Var = varb**
    - selected rows and columns as in **; Var = varb[1, 3, 5, 6]**.
    - numeric values, provided as a lower triangle, as in **; Var = 1.2, -3., 5.5**.

    You may use **; Covariance = specification** as above as a synonym.

- **; Fnj = ...** specifies the function to be analyzed. The full set of options for this part of the command are given in Chapters E14 and E66. For present purposes, any algebraic function of the estimates can be computed. Up to 50 functions can be defined. You may define names for the functions that will be used in the results table with **; Fnj = name = definition**. An example appears below.

- **; Keep = ...** is optional and specifies that only the specified functions are to be displayed and retained in the results. Use this when some results are intermediate. Thus, if you were to compute Fn1 ... Fn10, but you only were interested in Fn9 and Fn10, you might use **; Keep = 9,10** to discard the first eight, intermediate results.

- When you use **; Average ; Keep = name**, **WALD** creates two new variables in the data set, *waldfncn* which contains for each observation, the first function value in the list, and *waldfnse* which contains for each observation an estimate of the standard error.

R14.4.2 Results of the WALD Command

WALD produces a standard table of function values, standard errors, and so on. In the following example, we estimate a probit model, then compute the average partial effects. (This replicates the computations of **; Partial Effects** in the command and **PARTIAL EFFECTS**.)

```
PROBIT      ; Lhs = doctor ; Rhs = one,age,educ,hhninc,married $
WALD        ; Covariance = varb
            ; Parameters = b
            ; Labels = b1,b2,b3,b4,b5
            ; Fn1 = density    = Phi(b1+b2*age+b3*educ+b4*hhninc+b5*married)
            ; Fn2 = ape_age    = density * b2
            ; Fn3 = ape_educ   = density * b3
            ; Fn4 = ape_incm   = density * b4
            ; Fn5 = ape_marr   = Phi(b1+b2*age+b3*educ+b4*hhninc+b5)-
                                 Phi(b1+b2*age+b3*educ+b4*hhninc)
            ; Average $
```

```
-----------------------------------------------------------------------------
WALD procedure. Estimates and standard errors
for nonlinear functions and joint test of
nonlinear restrictions.
Wald Statistic             =  19344.31753
Prob. from Chi-squared[ 5] =       .00000
Functions of data are averaged over the obs.
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
WaldFcns|  Coefficient      Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
 DENSITY|     .36520***      .00291   125.45  .0000      .35950    .37091
 APE_AGE|     .00489***      .00064     7.65  .0000      .00364    .00615
APE_EDUC|    -.00431         .00312    -1.38  .1663    -.01042    .00179
APE_INCM|    -.14883***      .04505    -3.30  .0010    -.23713   -.06054
APE_MARR|     .03920**       .01735     2.26  .0239      .00519    .07321
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

WALD computes the value of each function you specify, its estimated standard error, the t ratio, and so on. It also computes the Wald statistic for the set of functions. The Wald statistic computed by this procedure is

$$W = \sum_{i=1}^{J} \sum_{j=1}^{J} \mathbf{Fn}i \times \mathbf{Fn}j \times [\mathbf{D}'\,\mathbf{VARB}\,\mathbf{D}]^{ij}$$

(The superscript indicates the element of an inverse matrix.) The statistic, itself, may be of no use to you, in which case, it can be ignored. Conversely, the specific functions may be the superfluous information, and the Wald statistic may be the only information that you need. In this case, the function output can be ignored. After the computation, matrices *waldfns*, *jacobian* and *varwald* will be the functions, **c**, the Jacobian, **D**, and the covariance matrix, **G**. Scalar *wald* will contain the Wald statistic, *W*.

R14.4.3 Recursive Functions

Functions in the **WALD** command may use previous functions. For example:

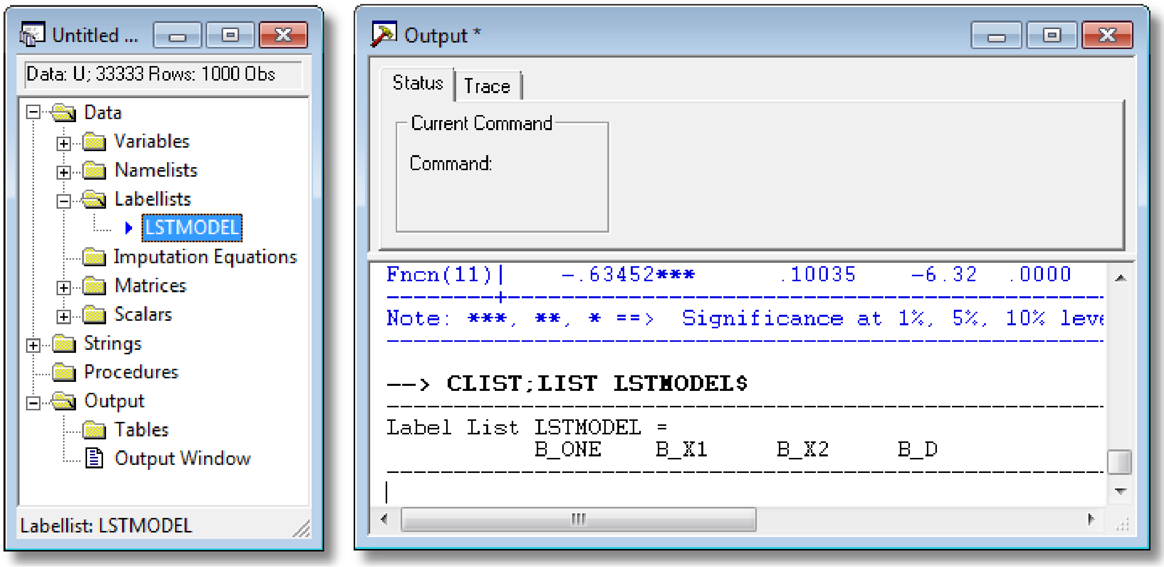
```
                  ; Fn1 = b'xbar
                  ; Fn2 = N01(Fn1)/Phi(Fn1)
                  ; Fn3 = s*(1 - Fn2*Fn1 - Fn2^2)
```

Do note, these *must* be defined recursively. In the preceding, Fn2 could not be defined as a function of Fn3. We used this capability in the first example above, where *density* is defined then used in three subsequent function definitions. This feature can be a particularly convenient and powerful aspect of the command. Consider the routine below for computing a marginal effect for a binary variable in a tobit model. The **WALD** command would be enormously complex and lengthy, if it were not specified recursively. Note, the final line, if included in the command, would indicate that only the 11th function is to be displayed and kept in the work areas.

```
SAMPLE      ; 1-1000 $
CREATE      ; x1 = Rnn(0,1) ; x2 = Rnn(0,1) ; d = Rnu(0,1) > .4 $
CREATE      ; ys = 2.5 + .5*x1 + .4 *x2 - .6*d + Rnn(0,2) $
CREATE      ; y = ys ; If(ys < 0) y = 0 ; If(ys > 5) y = 5 $
NAMELIST    ; x = one,x1,x2 $
CALC        ; l = 0 ; u = 5 ; kx = Col(x) $
TOBIT       ; Lhs = y ; Rhs = x,d ; Limits = l,u ; Parameters $
WALD        ; Labels = kx_b,alpha,v          ? all parameters
            ; Start = b ; Var = varb         ? includes sigma
            ; Fn1  = (L - b1'x)/v            ? (L - b'x1)/s
            ; Fn2  = (U - b1'x)/v            ? (U - b'x1)/s
            ; Fn3  = Phi(Fn1- alpha/v)       ? Phi[(L - b'x1)/s]
            ; Fn4  = Phi(Fn2 - alpha/v)      ? Phi[(U - b'x1)/s]
            ; Fn5  = N01(Fn1 - alpha/v)      ? N01[(L - b'x1)/s]
            ; Fn6  = N01(Fn2 - alpha/v)      ? N01[(U - b'x1)/s]
            ; Fn7  = Phi(Fn1)                ? Phi[(L - b'x0)/s]
            ; Fn8  = Phi(Fn2)                ? Phi[(U - b'x0)/s]
            ; Fn9  = N01(Fn1)                ? N01[(L - b'x0)/s]
            ; Fn10 = N01(Fn2)                ? N01[(U - b'x0)/s]
            ; Fn11 =
( Fn3 * L + (1 - Fn4) * U + (b1'x+alpha)* (Fn4 - Fn3) + v * (Fn5 - Fn6) )
- ( Fn7 * L + (1 - Fn8) * U + b1'x          * (Fn8 - Fn7) + v * (Fn9 - Fn10) )
            ; Average
            ; Keep = 11 $
```

R14.4.4 Application Based on the Last Model

The Last Model estimated produces a set of labels that you can use in the **WALD** command, with the parameter vector, *b*, and covariance matrix, *varb*. The labels that apply for the last model can be found in the project window. The labellist *lstmodel* is replaced each time you compute a new model. To see its current contents, double click the name in the project window, and a list will be placed in the output window. The example below is produced by the **TOBIT** command in the program above.



```
Fncn(11)|     -.63452***       .10035     -6.32   .0000
--------+----------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% leve
-------------------------------------------------------

--> CLIST;LIST LSTMODEL$
-------------------------------------------------------
Label List LSTMODEL =
           B_ONE    B_X1     B_X2     B_D
-------------------------------------------------------
```

Figure R14.1 Labellist from Last Model

You may use the last model labels with *b* and *varb* as defaults in your **WALD** command. The example in Section R14.4.2 can be simplified a bit as

```
PROBIT   ; Lhs = doctor ; Rhs = one,age,educ,hhninc,married $
WALD     ; Fn1 = bx = b_one +b_age*age + b_educ*educ + b_hhninc*hhninc
         ; Fn2 = density    = Phi(bx + b_married*married)
         ; Fn3 = ape_age    = density * b_age
         ; Fn4 = ape_educ   = density * b_educ
         ; Fn5 = ape_incm   = density * b_hhninc
         ; Fn6 = ape_marr   = Phi(bx +b5) - Phi(bx)
         ; Average $
```

Note that the construction above is invariant to how you order the variables in your model. Regardless of what else appears on your Rhs and in what order it appears, in the *Last Model* set, the name *b_age* will refer to the coefficient that multiplied *age* in the most recently estimated model.

R14.4.5 The Number of Parameters

In specifying a **WALD** command, you frequently need to know the number of coefficients in the coefficient vector. This will likely depend on the model you have fit or the procedure you have used to obtain the functions you are analyzing. When you fit a model, the number of coefficients in the coefficient vector that is produced is stored as a scalar named *kreg*. You can use *kreg* in your command. The preceding could have been

```
WALD        ; Start = b
            ; Var = varb
            ; Labels = kreg _ gamma
            ; Fn1 = gamma2 * N01(gamma1'xb)
            ; Fn2 = gamma3 * N01(gamma2'xb) $
```

The advantage in this form is that you can structure your procedures so that they are general and not dependent on a specific value or model. If you are not basing your computation on a previous model that stored *kreg*, you can still obtain the dimension that you need as follows: **CALC** provides the Row(*matrix*) function which returns the number of rows in a matrix. (The Col(*matrix*) function is also available.) Thus, the preceding could also be specified using

```
CALC        ; numbeta = Row(varb) $
WALD        ; Start = b
            ; Var = varb
          → ; Labels  = numbeta _ gamma
            ; Fn1 = gamma2 * N01(gamma1'xb)
            ; Fn2 = gamma3 * N01(gamma2'xb) $
```

NOTE: There is a dot product in the N01 function. The construction *b_one'xb* means compute the inner product of the coefficient vector with the variables in *xb*. The specific coefficient name *b_one* means start the coefficient vector with *b_one*. You might have (certainly incorrectly) used *b_gc'xb* which would compute *b_gc*one* + *b_ttme*gc* and a third term would be lost.

R14.4.6 Interdependent Sets of Functions

In some applications, instead of presenting a Wald statistic, the table will give the diagnostic, 'VC matrix for the functions is singular.' This is likely to happen when you compute a set of functions which are functionally dependent. This is deduced by the program attempting to invert the covariance matrix. Assuming the base covariance matrix, *varb*, is nonsingular, the matrix **G** will be singular if the derivatives matrix does not have full rank, in which case, we infer that the functions are functionally dependent. The application below, which displays the results of the program in Section R14.4.3, shows an example.

```
-----------------------------------------------------------------------------
Limited Dependent Variable Model - CENSORED
Dependent variable                      Y
Estimation criterion         -1852.69115
Estimation based on N =    1000, K =    5
Threshold values for the model:
Lower=      .0000     Upper=     5.0000
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Primary Index Equation for Model
Constant|    2.52695***      .10213    24.74  .0000     2.32678   2.72712
      X1|     .47633***      .06529     7.30  .0000      .34837    .60429
      X2|     .35063***      .06434     5.45  .0000      .22454    .47673
       D|    -.85204***      .13559    -6.28  .0000    -1.11780   -.58628
        |Disturbance standard deviation
   Sigma|    2.05626***      .05765    35.67  .0000     1.94328   2.16924
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
WALD procedure. Estimates and standard errors
for nonlinear functions and joint test of
nonlinear restrictions.
VC matrix for the functions is singular.
Standard errors are reported, but the
Wald statistic cannot be computed.
Functions of data are averaged over the obs.
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
WaldFcns|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
 Fncn(1)|   -1.22458***      .06032   -20.30  .0000    -1.34280  -1.10636
 Fncn(2)|    1.20702***      .06013    20.07  .0000     1.08916   1.32488
 Fncn(3)|     .21866***      .01436    15.23  .0000      .19052    .24680
 Fncn(4)|     .93997***      .00682   137.82  .0000      .92661    .95334
 Fncn(5)|     .28275***      .01077    26.25  .0000      .26164    .30387
 Fncn(6)|     .11431***      .01011    11.30  .0000      .09449    .13414
 Fncn(7)|     .12024***      .01153    10.43  .0000      .09764    .14283
 Fncn(8)|     .87638***      .01182    74.14  .0000      .85322    .89955
 Fncn(9)|     .19195***      .01297    14.80  .0000      .16653    .21736
Fncn(10)|     .19590***      .01304    15.03  .0000      .17035    .22145
Fncn(11)|    -.63452***      .10035    -6.32  .0000     -.83120   -.43784
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

If you are only interested in the functions, themselves, the diagnostic can be ignored. Nonetheless, you will receive estimates of the functions and standard errors for the individual components. The Wald statistic is of no interest anyway. The listing below shows the results of executing the preceding **WALD** command for the tobit model. As might be expected, the functions are dependent, and no Wald statistic is computed. In fact, for this application, the only quantity of interest is the last function, which gives the desired partial effect. As such, all rows save the last one in this table are ignored.

R14.4.7 Extracting Parts of a Model

It may be convenient to analyze just part of a model. Consider another example (a bit farfetched, we admit). We fit a probit model for *y* using 15 regressors including a constant. The hypotheses to be tested are

$$b_1b_4/(b_2 + b_5 + \exp(b_{11}) = 1$$

$$b_2 + b_3 = 4$$

The commands are

```
PROBIT     ; Lhs    = y ; Rhs = x1,x2,...,x15,one $
WALD       ; Labels = b1,b2,b3,b4,b5,b11
           ; Start  = b(1),b(2),b(3),b(4),b(5),b(11)
           ; Var    = varb[1,2,3,4,5,11]
           ; Fn1    = b1*b4/(b2+b5+ Exp(b11)) - 1
           ; Fn2    = b2 + b3 - 4 $
```

Note how the constant of each restriction is moved to the left hand side of the expression to conform to the convention $g_j(\mathbf{b}) = 0$. Also, as shown above, this procedure can be used for linear as well as nonlinear restrictions.

R14.4.8 Application to a Function of the Parameters

The **WALD** command can be used to analyze functions of parameters that do not involve the data. To use this feature, you may follow your model command with any number of commands of the form

```
WALD       ; Labels = a set of names for the parameters
           ; Parameters = the values of the estimates
           ; Covariance = the estimated covariance matrix
           ; Fn1 = function of model parameters
           ; Fn2 = function of model parameters
           ; ...  up to 50 functions  $
```

The command needs only to provide the desired function. Each **WALD** command may give up to 50 functions, but you may have as many **WALD** commands as you wish. To illustrate, we consider an extensive example based on Example 6.8 CES Production Function in Greene (2012, p. 167). One method of estimating the parameters of the CES production function,

$$\log y = \log\gamma - (\nu/\rho)\log[\delta k^{\rho} + (1-\delta)l^{\rho}]$$

is to regress *logy* on *one,x1* = log*k*, *x2* = log*l*, and *x3* = $\log^2(k/l)$. The coefficients thus obtained are labeled b_1, b_2, b_3, and b_4. The structural parameters are $\gamma = \exp(b_1)$, $\delta = b_2/(b_2+b_3)$, $\nu = (b_2+b_3)$, and $\rho = -2b_4(b_2+b_3)/(b_2b_3)$. The *b*s are estimated by ordinary least squares. We then use **WALD** to compute the structural parameters and estimate standard errors for these estimates. An application appears below.

```
IMPORT $
                    obs,  valueadd,   labor,    capital
                     1     657.29   162.31    279.99
                     2     935.93   214.43    542.50
                     3    1110.65   186.44    721.51
                     4    1200.89   245.83   1167.68
                     5    1052.68   211.40    811.77
                     6    3406.02   690.61   4558.02
                     7    2427.89   452.79   3069.91
                     8    4257.46   714.20   5585.01
                     9    1625.19   320.54   1618.75
                    10    1272.05   253.17   1562.08
                    11    1004.45   236.44    662.04
                    12     598.87   140.73    875.37
                    13     853.10   145.04   1696.98
                    14    1165.63   240.27   1078.79
                    15    1917.55   536.73   2109.34
                    16    9849.17  1564.83 13989.55
                    17    1088.27   214.62    884.24
                    18    8095.63   1083.10 9119.70
                    19    3175.39   521.74   5686.99
                    20    1653.38   304.85   1701.06
                    21    5159.31   835.69   5206.36
                    22    3378.40   284.00   3288.72
                    23     592.85   150.77    357.32
                    24    1601.98   259.91   2031.93
                    25    2065.85   497.60   2492.98
                    26    2293.87   275.20   1711.74
                    27     745.67   137.00    768.59

CREATE      ; y = Log(valueadd)
            ; x1 = Log(capital) ; x2 = Log(labor) ; x3 = (x1-x2)^2 $
REGRESS     ; Lhs = y ; Rhs = one,x1,x2,x3 $
WALD        ; Labels = b0,b1,b2,b3 ; Start = b ; Var = varb
            ; Fn1 = gamma = Exp(b0)
            ; Fn2 = delta    = b1 / (b1 + b2)
            ; Fn3 = nu       = b1 + b2
            ; Fn4 = ro       = -2 * b3 * (b1 + b2) / (b1 * b2) $
```

```
-----------------------------------------------------------------------------
Ordinary    least squares regression ............
LHS=Y       Mean                 =       7.44363
            Standard deviation   =        .76115
            No. of observations  =            27  Degrees of freedom
Regression  Sum of Squares       =       14.2614           3
Residual    Sum of Squares       =       .801802          23
Total       Sum of Squares       =       15.0632          26
            Standard error of e  =        .18671
Fit         R-squared            =        .94677  R-bar squared =   .93983
Model test  F[  3,    23]        =     136.36447  Prob F > F*   =   .00000
Diagnostic  Log likelihood       =       9.16451  Akaike I.C.   = -3.22043
            Restricted (b=0)     =     -30.43298
            Chi squared [  3]    =      79.19498  Prob C2 > C2* =   .00000
--------+--------------------------------------------------------------------
```

```
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       t    |t|>T*         Interval
--------+--------------------------------------------------------------------
Constant|    1.46773***       .40823     3.60  .0015      .66761   2.26784
      X1|    -.11150          .41620     -.27  .7912    -.92723    .70423
      X2|    1.10023**        .43422     2.53  .0186      .24917   1.95128
      X3|     .15225          .12734     1.20  .2440    -.09734    .40184
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
WALD procedure. Estimates and standard errors
for nonlinear functions and joint test of
nonlinear restrictions.
Wald Statistic              =  47548.33748
Prob. from Chi-squared[ 4] =       .00000
Functions are computed at means of variables
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
WaldFcns|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
   GAMMA|    4.33936**       1.77146     2.45  .0143      .86736   7.81135
   DELTA|    -.11277          .41944     -.27  .7880    -.93485    .70931
      NU|     .98872***       .06259    15.80  .0000      .86605   1.11139
      RO|    2.45416         8.08604      .30  .7615  -13.39419  18.30251
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The standard least squares results are followed by the output from the **WALD** command. The table contains the function values, computed at the parameter estimates, standard errors, and ratios of the function values to the standard errors. The standard errors are the square roots of the diagonal elements of the estimated asymptotic covariance matrix. This is computed using the ‘delta’ method, i.e., let

$$\text{Fn}j \;=\; c_j(\mathbf{b}),\, j = 1,\ldots,J$$

denote the *j*th function that you have specified, written only as a function of the full vector of parameter estimates, **b**. Let

$$\mathbf{g}_j = \;\partial c_j(\mathbf{b})/\partial\mathbf{b}'$$

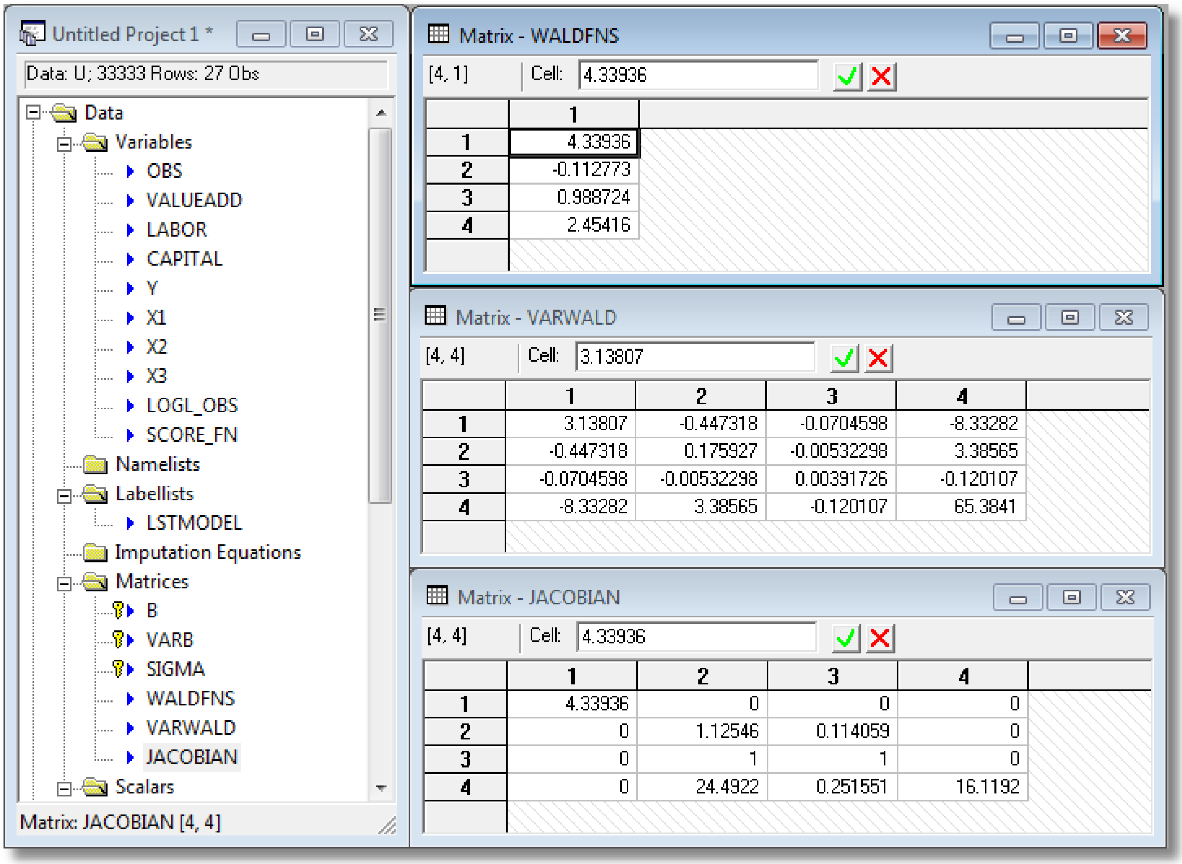
Note that $\mathbf{g}_j$ is a row vector which will usually contain some zeros, since the functions need not involve all of the parameter estimates. Let **G** denote the matrix whose *j*th row is $\mathbf{g}_j$, so **G** is $J\times kreg$. Then, the asymptotic covariance matrix for the set of functions is computed using

$$varwald \;=\; \mathbf{G}\times\mathbf{varb}\times\mathbf{G}'$$

After the functions are computed and reported, **WALD** retains three retrievable results:

Matrices:	*waldfns*	= a vector containing the J functions
	varwald	= a $J \times J$ estimated asymptotic covariance matrix
	jacobian	= a $J \times K$ matrix of derivatives of the functions with respect to the parameters
Scalar:	*wald*	= $\mathbf{c}'[\mathbf{varwald}]^{-1}\mathbf{c}$

Wald is the Wald (chi squared) statistic used to test the hypothesis that all functions are jointly zero. It is reported in the box header above the table of function values and standard errors. See Figure R14.2 for the results of the example above.


Untitled Project 1 *

Data: U; 33333 Rows: 27 Obs

```
Data
  Variables
    OBS
    VALUEADD
    LABOR
    CAPITAL
    Y
    X1
    X2
    X3
    LOGL_OBS
    SCORE_FN
  Namelists
  Labellists
    LSTMODEL
  Imputation Equations
  Matrices
    B
    VARB
    SIGMA
    WALDFNS
    VARWALD
    JACOBIAN
  Scalars
```

Matrix: JACOBIAN [4, 4]

Matrix - WALDFNS

[4, 1] Cell: 4.33936

	1
1	4.33936
2	-0.112773
3	0.988724
4	2.45416

Matrix - VARWALD

[4, 4] Cell: 3.13807

	1	2	3	4
1	3.13807	-0.447318	-0.0704598	-8.33282
2	-0.447318	0.175927	-0.00532298	3.38565
3	-0.0704598	-0.00532298	0.00391726	-0.120107
4	-8.33282	3.38565	-0.120107	65.3841

Matrix - JACOBIAN

[4, 4] Cell: 4.33936

	1	2	3	4
1	4.33936	0	0	0
2	0	1.12546	0.114059	0
3	0	1	1	0
4	0	24.4922	0.251551	16.1192

Figure R14.2 Results from WALD Procedure

R14.4.9 Application to a Complex Nonlinear Function

As might be evident, the **WALD** command is more general than we have suggested, and it can be an extremely powerful time saver. An example which normally involves a large amount of computation is the predicted value for the Box-Cox regression. (The model and the following example are described in more detail in Chapter E13.) The prediction for the Box-Cox model when the transformation is applied to both Lhs and Rhs variables is

$$\hat{y} = \left[\lambda\left(\sum k\, b_k ((x_k^{\lambda} - 1)/\lambda)\right) + 1\right]^{1/\lambda}.$$

If λ is an estimated parameter whose variation must be included in the computation of the forecast standard error, this becomes an exceedingly complex computation. With **WALD**, the computation can be done as follows, where we use the means of two regressors for the forecast:

```
BOXCOX      ; Lhs = y
            ; Rhs = x1,x2,one
            ; Model = 3 ; Lambda = ... ; Par $
CALC        ; u1 = Xbr(x1) ; u2 = Xbr(x2) $
WALD        ; Labels  = b1,b2,a,L
            ; Start = b(1),b(2),b(3),b(4)
            ; Var = varb[1,2,3,4]
            ; Fn1 = (L*(b1*x1@L + b2*x2@L + a) + 1)^(1/L) $
```

Using **; Par** with the command saves the ancillary parameters, λ and s^2 in *b* and *varb*.

R15: Retrievable Results

R15.1 Introduction

When you use *LIMDEP* (or any other program) to compute estimates of parameters or tables of results, you will need to be able to retrieve the results of estimation to do subsequent calculations. Otherwise, you are limited to what the software provides in your tables of results in what you can do with those results. To consider an example, it is common after estimation of multinomial choice models to compute a 'willingness to pay' result using a ratio of the model parameters. In the following example,

```
CLOGIT      ; Lhs = mode
            ; Choices = air,train,bus,car
            ; Rhs = gc,invt,hinca,one $
```

the outcome variable is the choice of travel mode and the characteristics are a cost variable, *gc*, and the time spent in the journey, *invt*. *Hinca* is an income variable that applies only in the air choice. After estimation of β_{gc}, β_{invt} and the other parameters, one can measure the willingness to pay (WTP) for a shorter journey by the ratio, $(-\beta_{invt}/-\beta_{gc})$. Estimation results are shown in Figure R15.1. At this point, in order to compute the WTP, it is necessary to take out pen and paper and a hand calculator. In addition, computing a standard error for the result will require the covariance matrix (and a calculator) to employ, at considerable inconvenience, the delta method. In order to avoid this inconvenience, it is necessary to be able to 'retrieve' the results of estimation in a way that they can be manipulated using program instructions. (A menu of 'post estimation' features is helpful, but will be insufficient unless the program has every possible calculation you might want in its menu.)

We provide the calculator for you with the **CALC** command described in Chapter R17. So to begin, you could get the result you need with

```
CALC        ; wtp = .00269 / .00861 $
```

which will produce the result 0.3127943. This obviously does not solve the problem, because a different specification requires a new pair of values. What is required is to be able to insert the values of the parameters as names of something that the program has computed. If we can retrieve the results of estimation in something that has a name, we can manipulate the names to get the result we need. Estimation results in *LIMDEP* are always retrievable in a set of named entities (scalars, matrices, variables, etc.) To continue our example, the parameter vector computed by every estimation command in *LIMDEP* is saved as a matrix named *b* which can be accessed as a matrix or one element at a time with the calculator. Thus, the desired WTP measure is computed by

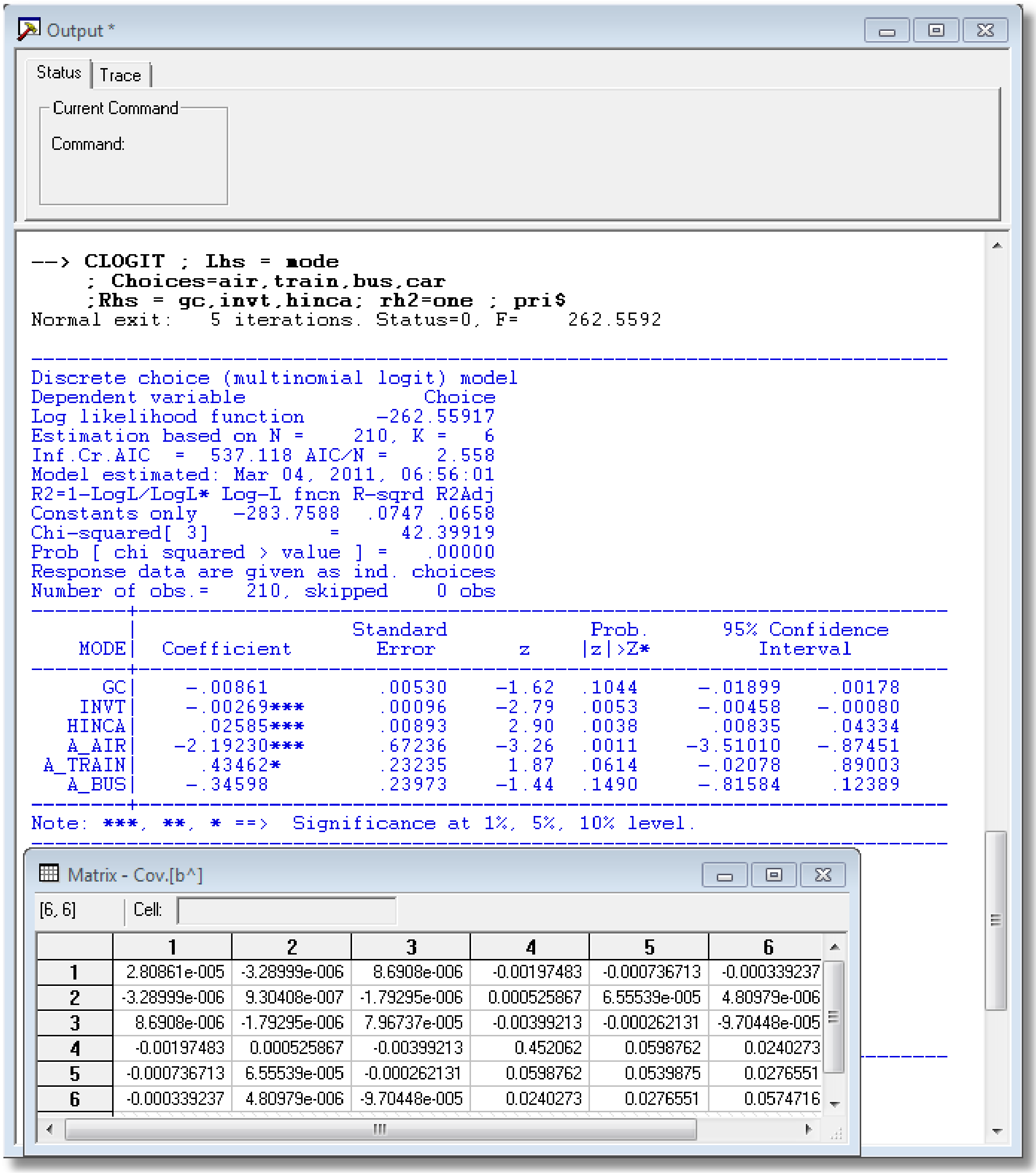
```
CALC        ; List ; wtp =  -b(2)/(-b(1)) $
```

```
[CALC] wtp  = .3127943
```

There is a minor additional inconvenience in this computation in that it insists that the two coefficients be the first and second in the model. But, this is not strictly necessary either because the **CLOGIT** command also automatically saves the names of the parameters. The **CALC** command below is completely generic, and will find the desired result as long as *invt* and *gc* appear in the model.

CALC ; List ; wtp = b_invt/b_gc $

```
[CALC] wtp  =   .3127943
```



```
--> CLOGIT ; Lhs = mode
    ; Choices=air,train,bus,car
    ;Rhs = gc,invt,hinca; rh2=one ; pri$
Normal exit:   5 iterations. Status=0, F=    262.5592

-----------------------------------------------------------------------------
Discrete choice (multinomial logit) model
Dependent variable               Choice
Log likelihood function      -262.55917
Estimation based on N =    210, K =    6
Inf.Cr.AIC  =  537.118 AIC/N =    2.558
Model estimated: Mar 04, 2011, 06:56:01
R2=1-LogL/LogL* Log-L fncn R-sqrd R2Adj
Constants only   -283.7588  .0747  .0658
Chi-squared[ 3]          =    42.39919
Prob [ chi squared > value ] =   .00000
Response data are given as ind. choices
Number of obs.=   210, skipped    0 obs
--------+--------------------------------------------------------------------
        |                 Standard            Prob.       95% Confidence
    MODE|  Coefficient      Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
      GC|   -.00861        .00530    -1.62  .1044     -.01899     .00178
    INVT|   -.00269***     .00096    -2.79  .0053     -.00458    -.00080
   HINCA|    .02585***     .00893     2.90  .0038      .00835     .04334
   A_AIR|  -2.19230***     .67236    -3.26  .0011    -3.51010    -.87451
 A_TRAIN|    .43462*       .23235     1.87  .0614     -.02078     .89003
   A_BUS|   -.34598        .23973    -1.44  .1490     -.81584     .12389
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

	1	2	3	4	5	6
1	2.80861e-005	-3.28999e-006	8.6908e-006	-0.00197483	-0.000736713	-0.000339237
2	-3.28999e-006	9.30408e-007	-1.79295e-006	0.000525867	6.55539e-005	4.80979e-006
3	8.6908e-006	-1.79295e-006	7.96737e-005	-0.00399213	-0.000262131	-9.70448e-005
4	-0.00197483	0.000525867	-0.00399213	0.452062	0.0598762	0.0240273
5	-0.000736713	6.55539e-005	-0.000262131	0.0598762	0.0539875	0.0276551
6	-0.000339237	4.80979e-006	-9.70448e-005	0.0240273	0.0276551	0.0574716

Figure R15.1 Estimated Travel Mode Choice Model

Suppose we wished to test the hypothesis that travelers did not value their travel time. That would entail a test of the hypothesis that WTP equals zero. To carry out the test, we could, in principle, just test the hypothesis that the coefficient in the numerator of WTP equals zero (which we would reject based on the estimation results in Figure R15.1). Alternatively, we could analyze the WTP result itself. To apply the delta method to that, we would require the covariance matrix, which is also retrievable. The **WALD** command below shows how to make use of the saved matrices and *varb*.

```
WALD        ; Labels = bgc,binvt,chinca,cair,ctrain,cbus
            ; Parameters = b
            ; Covariance  = varb
            ; Fn1 = wtp = -binvt/-bgc $
```

```
-----------------------------------------------------------------------------
WALD procedure. Estimates and standard errors
for nonlinear functions and joint test of
nonlinear restrictions.
Wald Statistic              =      1.26306
Prob. from Chi-squared[ 1] =       .26107
Functions are computed at means of variables
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
WaldFcns|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     WTP|     .31279         .27832     1.12  .2611     -.23271    .85830
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Estimation results produced by *LIMDEP* are always retrievable, and the program command language provides many convenient ways to manipulate those results. As shown above, the main tools you will use to manipulate the retrievable results are **CALC**, **MATRIX**, **CREATE** and **WALD**. The **PARTIAL EFFECTS**, **SIMULATE** and **DECOMPOSE** commands also use this aspect of estimation. There are also several numerical analysis tools described in Chapter E68 that can retrieve and use previous estimation results.

R15.2 Retrievable Results

When you estimate a model, the estimation results are displayed on the screen in the output window. In addition, each model produces a number of results which are saved automatically and can be used in subsequent procedures and commands. Retrievable results generally appear in four locations in the project, as variables, matrices, scalars and labellists.

The **CLOGIT** command above shows an example. After the model is estimated, scalars named *nreg*, *kreg*, and *logl* are created and set equal to the number of observations, number of coefficients estimated, and the log likelihood for the model, respectively. For another, after you give a **REGRESS** command, the scalar *rsqrd* is thereafter equal to the R^2 from that regression. You can retrieve these and use them in later commands. For example,

```
REGRESS     ; ... $
CALC        ; f = rsqrd/(kreg-1) / ((1 - rsqrd)/(nreg - kreg)) $
```

computes a standard F statistic.

Although the calculator has 100 cells, the first 14 are ‘read only’ in the sense that *LIMDEP* reserves them for estimation results. You may use these scalars in your calculations, or in other commands (see the example above), but you may not change them. (The one named *rho* may be changed.) Likewise, the first three matrices are reserved by the program for ‘read only’ purposes. The read only scalars are

ssqrd, rsqrd, s, sumsqdef, degfrdm, ybar, sy, kreg, nreg, logl, exitcode

and two whose names and contents will depend on the model just estimated. The names used for these will be given with the specific model descriptions. At any time, the names of the read only scalars are marked in the project window with the 🔑 symbol to indicate that these names are ‘locked.’ Figure R15.2 illustrates. This shows the setup of the project window after the CLOGIT example developed above.

A parameter vector is automatically retained in a matrix named *b*. The program will also save the estimated asymptotic covariance matrix and name it *varb*. The reserved matrices are thus *b* and *varb*, with a third occasionally used and renamed. The third, protected matrix name will depend on the model estimated. A few examples are:

mu	created by **ORDERED PROBIT**,
sigma	created by **SURE** and **3SLS**,
pacf	created by **IDENTIFY**.

When a model is fit by maximum likelihood, a variable named *logl_obs* is created. The variable contains the contribution of each observation to the log likelihood that was maximized. In the CLOGIT example above, the log likelihood function reported in the results is -262.55917. We could locate this result with

```
CALC            ; List ; Sum(logl_obs) $
```

```
[CALC] *Result*=   -262.5591744
```

In many single equation, single index models, such as the probit model or the linear regression model, the derivative of the log likelihood function with respect to the **β** in the index function **β′x** takes the form

$$\partial \log L / \partial \boldsymbol{\beta} = \sum_{i=1}^{n} g(\boldsymbol{\beta}, \boldsymbol{\theta}, \mathbf{x}_i, y_i)\mathbf{x}_i,$$

where $\mathbf{x}_i$ is the set of independent variables, y_i is the dependent variable, **θ** is a vector of ancillary parameters, and **β** is the vector of coefficients on $\mathbf{x}_i$ in the model. The function *g*(.) is often called the ‘score function.’ It is also the derivative of the log likelihood function with respect to the constant term. This is a residual-like function. It is the ‘generalized residual’ defined in Chesher and Irish (1987). For examples, $g_i = \varepsilon_i / \sigma^2$ in the linear regression model when the disturbances are normally distributed, and $g_i = (2y_i\text{-}1)\,\phi(\boldsymbol{\beta}'\mathbf{x}_i) / \Phi[(2y_i\text{-}1)\boldsymbol{\beta}'\mathbf{x}_i]$ in the probit model. (Note that this is the sample selection correction term in Heckman’s (1979) two step sample selection estimator.) This variable will be saved in the data area as *score_fn*. (The variable is created for every model fit by ML. When there is no natural index function, for example in the bivariate probit model which has two index functions, *score_fn* will be filled with missing values.)

All estimators set at least some of these matrices, variables and scalars. In the case of the scalars, those not saved by the estimator are set to zero. For example, the **PROBIT** estimator does not save *rsqrd*. Matrices are simply left unchanged. So, for example, if you estimate a fixed effects model, which creates the third matrix and calls it *alphafe*, then estimate a probit model which only computes *b* and *varb*, *alphafe* will still be defined.

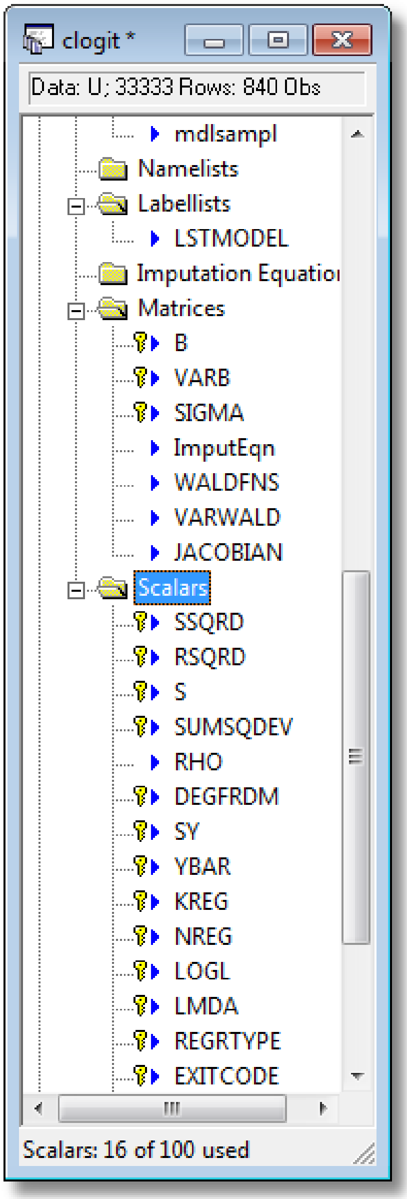


Figure R15.2 Project Window

TIP: Each time you estimate a model, the contents of *b*, *varb*, and the scalars are replaced. If you do not want to lose the results, retain them by copying them into a different matrix or scalar. For example, the following computes a Wald test statistic for the hypothesis that the slope vector in a regression is the same for two groups (a Chow test of sorts):

```
REGRESS     ; Lhs = y ; Rhs = ...  ; If [male = 1] $
MATRIX      ; bmale  =  b ; vmale = varb $
REGRESS     ; Lhs = y ; Rhs = ...  ; If [female = 1] $
MATRIX      ; bfemale = b ; vfemale = varb $
MATRIX      ; d = bmale - bfemale
            ; waldstat = d' * Nvsm(vmale, vfemale) * d $
```

The matrix results saved automatically in *b* and *varb* are, typically, a slope vector, **b**, and the estimated asymptotic covariance matrix of the estimator, from an index function model. For example, when you estimate a tobit model, the estimates and asymptotic covariance matrix are

$$\begin{bmatrix}\boldsymbol{\beta}\\ \sigma\end{bmatrix} \text{ and } \begin{bmatrix}\mathbf{V}_{\beta\beta} & \mathbf{V}_{\beta\sigma}\\ \mathbf{V}_{\sigma\beta} & \mathbf{V}_{\sigma\sigma}\end{bmatrix}.$$

The results kept are $\boldsymbol{\beta}$, in *b*, $\mathbf{V}_{\beta\beta}$ in *varb*, and σ in a scalar named *s*. The other parts of the asymptotic covariance matrix are generally discarded. We call the additional parameters, such as *s*, the *ancillary* parameters in the model. Most of the models that *LIMDEP* estimates contain one or two ancillary parameters. These are generally handled as in this example; the slope vector is retained as *b*, the ancillary parameters are kept as named scalars, and the parts of the covariance matrix that apply to them are discarded.

In some applications, you may want the full parameter vector and covariance matrix. You can retain these, instead of just the submatrices listed above, by adding the specification

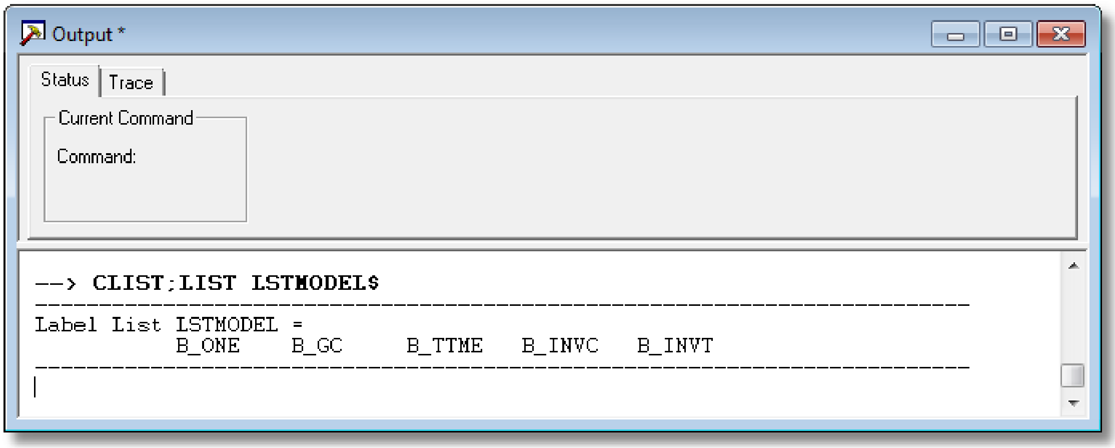
```
            ; Parameters
```

(or, just **; Par**) to your model command. (Note, for example, the computation of marginal effects for a dummy variable in a tobit model developed in Section R14.4.3.) Without this specification, the saved results are exactly as described above. The specific parameters saved by each command are listed with the model application in the chapters to follow. You will find an example of the use of this parameter setting in the program for marginal effects for a binary variable in the tobit model, which is in the previous section.

Finally, there is occasional use, particularly in the **WALD** command, for the labels of the parameters of the last model. Note in the project window in Figure R15.2, the labellist *lstmodel* is shown in the Labellists category. By double clicking this item, we can see in the output window the list of labels that have been assigned to the parameters fit by the previous model command. For example, after the command

```
LOGIT       ; Lhs = mode ; Rhs = one,gc,ttme,invc,invt $
```

Double clicking the *lstmodel* name in the project window displays the list shown in Figure R15.3.



```
--> CLIST;LIST LSTMODEL$
-------------------------------------------------------------------------
Label List LSTMODEL =
           B_ONE    B_GC     B_TTME   B_INVC   B_INVT
-------------------------------------------------------------------------
```

Figure R15.3 Last Model Labellist

R16: Using Matrix Algebra

R16.1 Introduction

The data manipulation and estimation programs described in the chapters to follow are part of *LIMDEP*'s general package for data analysis. The **MATRIX**, **CREATE**, and **CALCULATE** commands provide most of the additional tools. By using the **NAMELIST**, **SAMPLE, REJECT, INCLUDE, PERIOD,** and **DRAW** commands, you can arbitrarily define as many data matrices as you want. Simple, compact procedures using **MATRIX** commands will then allow you to obtain covariance and correlation matrices, condition numbers, and so on. More involved procedures can be used in conjunction with the other commands to program new, possibly iterative, estimators, or to obtain complicated partial effects or covariance matrices for two step estimators.

To introduce this extensive set of tools and to illustrate its flexibility, we will present several examples. The rest of the chapter will provide some technical results on matrix algebra and material on how to use **MATRIX** to manipulate matrices. (Most of these examples are hardwired procedures in *LIMDEP*, so the matrix programs are only illustrative.)

Example 1. Restricted Least Squares

In the linear regression model, $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, the linear least squares coefficient vector, $\mathbf{b}^*$, and its asymptotic covariance matrix, computed subject to the set of linear restrictions $\mathbf{Rb}^* = \mathbf{q}$ are

$$\mathbf{b}^* = \mathbf{b} - (\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}(\mathbf{Rb}-\mathbf{q}),$$

where $$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

and $$\text{Est.Asy.Var}[\mathbf{b}^*] = s^2(\mathbf{X}'\mathbf{X})^{-1} - s^2(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}.$$

First, define the **X** matrix, columns then rows. We assume the dependent variable is *y*.

```
NAMELIST    ; x = ... $ This defines the columns
CREATE      ; y = the dependent variable $
SAMPLE      ; ... as appropriate $ This defines the rows
```

Next, define **R** and **q**. This varies by the application. Get the inverse of **X'X** now, for convenience.

```
MATRIX      ; r = ... ; q = ... ; xxi = <x'x> $
```

Compute the unrestricted least squares coefficients and the discrepancy vector.

```
MATRIX      ; bu = xxi * x'y ; d = r * bu - q $
```

Compute the restricted least squares estimates and the sum of squared deviations.

```
MATRIX      ; br = bu - xxi * r' * Iprd(r,xxi,r') * d $
CREATE      ; u = y - x'br $
```

Compute the disturbance variance estimator.

```
CALC        ; s2 = (1/(n-Col(x)+Row(r))) * u'u $
```

Compute the covariance matrix, then display the results.

```
MATRIX     ; vr = s2 * xxi - s2 * xxi * r' * Iprd(r,xxi,r') * r * xxi $
DISPLAY    ; Parameters = br ; Covariance = vr ; Labels = x
           ; Title = Restricted Least Squares Estimates $
```

(The **MATRIX** function, **Stat(br,vr,x)** produces the same output as **DISPLAY**.)

The preceding gives the textbook formula for obtaining the restricted least squares coefficient vector when **X′X** is nonsingular. For the case in which there is multicollinearity, but the restrictions bring the problem up to full rank, the preceding is inadequate. (See Greene and Seaks (1991).) The general solution to the restricted least squares problem is provided by the partitioned matrix equation:

$$\begin{bmatrix} \mathbf{X'X} & \mathbf{R'} \\ \mathbf{R} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{b*} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{X'y} \\ \mathbf{q} \end{pmatrix}.$$

If the matrix in brackets can be inverted, then the restricted least squares solution is obtained along with the vector of Lagrange multipliers, $\boldsymbol{\lambda}$. The estimated asymptotic covariance matrix will be the estimate of σ^2 times the upper left block of the inverse. If **X′X** has full rank, this coincides with the solution above. A routine for this more general computation is

```
MATRIX     ; xx  = x'x ; xy = x'y ; r = ... ; q = ... $
CALC       ; k = Col(x) ; j = Row(r) $
MATRIX     ; zero  = Init(j, j, 0)
           ; a = [xx / r,zero] ? Shorthand for symmetric partitioned matrix
           ; v = [xy / q]
           ; ai = Ginv(a) ; b_l = ai * v
           ; br  = b_l(1:k) ; vr = ai(1:k, 1:k) $
CREATE     ; u = y - x'br $
MATRIX     ; vr =  { u'u  / (n-k+j) } * vr
           ; Stat(br,vr,x) $
```

Example 2. Poisson Model with a Fixed Value Restriction

In order to compute a Poisson regression model with different exposure rates, the solution is to enter the log of the exposure variable in the model with a fixed coefficient equal to 1.0. The restriction can imposed with the **; Rst = option** in the **POISSON** command (or with **; Exposure = variable name**). The following is an iterative procedure that would compute the same results for this application. It is necessary to set up the matrix procedure first. We isolate the last element of **b** to obtain the vector *beta*; *delta* is the update vector, initialized at zero, so the first iteration uses the starting values.

```
NAMELIST   ; x = one, … $
CREATE     ; y = the dependent variable
CREATE     ; logt  = the log of the exposure variable $
POISSON    ; Lhs  = y ; Rhs = x,logt $ For now, ignore the constraint.
CALC       ; k = Col(x) ; conv = 1 $
MATRIX     ; beta = b(1:k) ; delta = [k|0] $
```

This is the iterative procedure:

1. Update *beta*.
2. Compute the expected values, imposing slope on Log(months) = 1 and residuals.
3. Exit rule: *conv* must be initialized above because it is checked at entry to the iteration, not at exit. I.e., the execute procedure checks *conv* first, then decides whether or not to execute the procedure again. So, we make sure the check fails the first time it is tested.

```
PROCEDURE $
MATRIX      ; beta = beta - delta $
CREATE      ; ey = Exp(x'beta +  logt)
            ; uy = ey - y $
MATRIX      ; g = x'uy              ? first derivatives vector
            ; h = <x'[ey]x>         ? negative of second derivatives matrix
            ; delta = h * g         $ update vector
CALC        ; List ; conv = g'delta  $ This is the scale free convergence measure.
ENDPROCEDURE $
```

Execute the procedure until convergence, then display final results.

```
EXECUTE     ; until conv < .00001 $
MATRIX      ; Stat(beta,h,x) $
```

This program produces a trace of the iterations followed by the statistical output:

```
[CALC] CONV    =  46194.6187903
[CALC] CONV    =  10427.4732874
[CALC] CONV    =   1091.7873932
[CALC] CONV    =     18.9173685
[CALC] CONV    =       .0067992
[CALC] CONV    =       .0000000
CONV<.00001

-----------------------------------------------------------------------------
Number of observations in current sample =    4481
Number of parameters computed here       =       4
Number of degrees of freedom             =    4477
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
  Matrix|  Coefficient     Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    -.99753***      .06572   -15.18  .0000    -1.12634   -.86871
     AGE|     .02050***      .00081    25.44  .0000      .01893    .02208
    EDUC|    -.04218***      .00412   -10.23  .0000     -.05025   -.03410
 MARRIED|    -.11921***      .02070    -5.76  .0000     -.15979   -.07863
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Example 3. Plotting an Estimation Criterion Function

In some applications, a direct optimization of a criterion function with a gradient method, such as Newton's method used for the Poisson model in Example 2, is not feasible. When the search is for a single parameter, an alternative is to fix the parameter in a grid of values and plot the values of the criterion function over that grid to search for the optimum. The lag weight in a geometric lag model is an example. The following routine uses nonlinear least squares. The sum of squares is evaluated over an interval of values for λ, then plotted against those values. The regression model ultimately deduced is estimated at the end of the routine. (Note that for this specific application, the covariance matrix listed in that output would not be correct, as it does not correctly adjust for the use of nonlinear least squares.)

First, define matrices to keep the parameters and sums of squares.

```
MATRIX      ; ee = [40|0] ; l = ee $
```

Set up the initial value of the subscript and estimation criterion.

```
CALC        ; i = 1 ; eemin = 9999999 ; best = 0 $
```

Define the procedure to estimate the regression and keep the results. We also retain the optimal value of *lambda* and its sum of squares.

```
PROCEDURE $
CREATE      ; If( _obsno = 1 ) | z = lambda ; xstar = x
            ; (Else) | z = lambda * z[-1] ; xstar = x + lambda * xstar[-1] $
REGRESS     ; Lhs = y ; Rhs = one,z,xstar ; Quiet $
MATRIX      ; ee(i) = sumsqdev ? Note the use of subscripts for the matrix elements.
            ; l(i) = lambda
            ; If [sumsqdev < eemin] ; best = lambda ; eemin = sumsqdev $
CALC        ; List ; i = i + 1 $
ENDPROCEDURE $
```

Execute the procedure for the 40 values of *lambda*, then plot the values.

```
EXECUTE     ; lambda = .01 (.025) .99 $
MPLOT       ; Lhs  = l ; Rhs = ee ; Fill ; Endpoints = 0,1 $
CALC        ; i = 1 ; List ; best $
EXECUTE     ; Lambda = best $
REGRESS     ; Lhs  = y ; Rhs = one,z,xstar $
```

Notice the use of the scalars in the **CREATE** commands and as subscripts for the matrices.

Example 4. Canonical Correlations

Variables y_1, ..., y_L and x_1,...,x_K are arranged in $n \times L$ and $n \times K$ data matrices **Y** and **X**. The canonical variates (y^*,x^*) are those M = Min(L,K) pairs of linear functions of **Y** and **X** that have maximum correlation chosen so that all variables with unequal subscripts are uncorrelated. The canonical correlations are their pairwise correlation coefficients, $r1^*$... rM^*, ordered from largest to smallest. There are several ways to compute canonical correlations and canonical variates. The following has the useful virtue that it involves only symmetric matrices. This simplifies the computations because we need to compute characteristic roots, and decomposing symmetric matrices is simpler in this regard. Define the matrix product

$$\mathbf{R} = \mathbf{R}_{yy}^{-½}\mathbf{R}_{yx}\mathbf{R}_{xx}^{-1}\mathbf{R}_{xy}\mathbf{R}_{yy}^{-½},$$

where $\mathbf{R}_{ij}$, $i,j = x,y$, is a sample correlation matrix. The characteristic roots of **R** are the squared canonical correlation coefficients. The ordered canonical variates are contained in

$$\mathbf{y}^* = \mathbf{YRC} = \mathbf{YQ},$$

where the mth column of **C** is the characteristic vector of **R** corresponding to the mth largest nonzero root, and

$$\mathbf{x}^* = \mathbf{XR}_{xx}^{-1}\mathbf{R}_{xy}\mathbf{Q} = \mathbf{XV}.$$

The columns of **Q** are normalized to have unit length. The following program computes the canonical correlations and the coefficients of the canonical variates. It is assumed that *y* and *x* are namelists defining the sets of variables and that *y* does not have more variables than *x***,** so that *M* is the number of columns in *x*. Also, since this is based on correlations from the outset, neither *x* nor *y* may contain a column of ones (i.e., a constant term).

The following demonstrates how using matrix functions compresses large amounts of computation in small numbers of commands. We have shown the computations with a sampling experiment that you can use to demonstrate the procedures. Some of the intermediate output from the procedure is omitted. The random values for the experiment are produced by generating four independent columns of draws from the standard normal to constitute **X** and three for **Y**. Then, a linear combination of the seven random variables, with random coefficients, is used to induce some intercorrelation of the variables.

Define the variables in the computations.

```
SAMPLE      ; 1-1000$
CALC        ; Ran(12347) $  Set the seed so you can replicate the experiment.
CREATE      ; x1= Rnn(0,1) ; x2 = Rnn(0,1) ; x3 = Rnn(0,1) ; x4 = Rnn(0,1) $
CREATE      ; y1= Rnn(0,1) ; y2 = Rnn(0,1) ; y3 = Rnn(0,1) $
NAMELIST    ; x  = x1,x2,x3,x4 ; y = y1,y2,y3
            ; z  = x,y $            Note, z is all seven variables.
MATRIX      ; w = Rndm(7,7) $       Matrix of random numbers
MATRIX      ; z  = Xmlt(w) $        Mixture of all seven variables
```

Compute the simple correlations and cross correlations.

```
NAMELIST    ; x = list of variables ; y = list of variables $
MATRIX      ; List ; rxx = Xcor(x)  ; ryy = Xcor(y)  ; rxy = Xcor(x,y)  $
```

These **MATRIX** commands do the following:

1. Compute and display the sample canonical correlations.
2. Column *i* of **Q** is the coefficients of variate *y**(*i*).
3. Column *i* of **V** is the coefficients of variate *x**(*i*).
4. Squared canonical correlations are the diagonals of **R**.

```
MATRIX      ; List ; rr = Isqr(ryy) * rxy' * <rxx> * rxy * Isqr(ryy) $
MATRIX      ; List ; r = Root(rr) ; r = Diag(r) $
MATRIX      ; List ; q = r * Cvec(rr) ; norm = Diag(q'q) $
MATRIX      ; List ; q = q * Isqr(norm) $
MATRIX      ; List ; v = <rxx> * rxy * q $
```

```
Correlation Matrix for Listed Variables
--------+-----------------------------------------------------------------
        |      X1        X2        X3        X4
--------+-----------------------------------------------------------------
      X1| 1.00000    .66647    .13895    .70138
      X2|  .66647   1.00000  -.27401    .25449
      X3|  .13895  -.27401   1.00000  -.05935
      X4|  .70138    .25449  -.05935   1.00000

Correlation Matrix for Listed Variables
--------+-----------------------------------------------------------------
        |      Y1        Y2        Y3
--------+-----------------------------------------------------------------
      Y1| 1.00000  -.00686  -.15644
      Y2| -.00686  1.00000    .06252
      Y3| -.15644    .06252  1.00000

Correlation Matrix for Listed Variables
--------+-----------------------------------------------------------------
        |      Y1        Y2        Y3
--------+-----------------------------------------------------------------
      X1| -.39031    .77714  -.07281
      X2|  .00490    .74738  -.18534
      X3| -.17284    .13123  -.08101
      X4| -.03555    .55156    .10582

RR      |             1             2             3
--------+------------------------------------------
       1|       .724403      .00308662      .0609734
       2|     .00308662        .838933      -.159083
       3|      .0609734       -.159083      .0940829
```

```
R        |              1
---------+--------------
        1|        .872078
        2|        .729263
        3|       .0560780

R        |              1              2              3
---------+------------------------------------------
        1|        .872078        .000000        .000000
        2|        .000000        .729263        .000000
        3|        .000000        .000000       .0560780

Q        |              1              2              3
---------+------------------------------------------
        1|      -.0559187        .866740      -.0784475
        2|        .712281       .0590669        .144884
        3|      -.0114815      .00421263       .0547281

NORM     |              1              2              3
---------+------------------------------------------
        1|        .510604        .000000        .000000
        2|        .000000        .754745        .000000
        3|        .000000        .000000       .0301406

Q        |              1              2              3
---------+------------------------------------------
        1|      -.0782555        .997674       -.451859
        2|        .996804       .0679898        .834536
        3|      -.0160678      .00484900        .315235

V        |              1              2              3
---------+------------------------------------------
        1|       -.104287       -2.17004        .766933
        2|        .817439        1.35068       .0842229
        3|        .410255        .579855       .0759146
        4|        .440346        1.21526      -.0451223
```

Example 5. Discriminant Analysis

A data matrix, **X**, with n rows consists of two submatrices, $\mathbf{X}_1$ in n_1 rows and $\mathbf{X}_2$ in the other n_2 rows. ‘Discriminant analysis’ prescribes the following classification rule for an observation, $\mathbf{x}_i$: Classify in Group 1 if

$$\mathbf{x}_{i'}\mathbf{d} > ½ \left[\bar{\mathbf{x}}_1 + \bar{\mathbf{x}}_2\right]'\mathbf{d}$$

where $$\mathbf{d} = \mathbf{V}^{-1}\left[\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_2\right]$$

and $$\mathbf{V} = [\mathbf{X_1}'\mathbf{X}_1 + \mathbf{X_1}'\mathbf{X_1}]/(n_1+n_2\text{-}2).$$

The data used to compute **V** are in deviations from the respective subsample means. A set of commands that could be used for this computation are as follows. (This calculation, with numerous extensions, is provided by the model command **CLASSIFY**.)

These two commands are specific to the application. Define the data matrix.

```
NAMELIST    ; x   = list of names not including one $
CREATE      ; d1 = group 1 indicator variable
            ; d2 = 1 - d1 ? Group 2 indicator
            ; group = d1 + 2 * d2 $ Takes values 1 and 2
```

Obtain the means and variances to obtain **V**, then compute **D**.

```
MATRIX      ; xbar1 = <d1’1> * x’d1
            ; xbar2 = <d2’1> * x’d2
            ; s1 = {d1’d1 - 1} * Xvcm(x,d1)
            ; s2 = {d2’d2 - 1) * Xvcm(x,d2)
            ; v = {1/(n-2)} * Msum(s1,s2)
            ; d =  <v> * Mdif(xbar1,xbar2) $
```

Compute the classification variable. Create a binary variable = 1 if classified in group 1 and 0 if in group 2. (2 - this variable) produces the 1s and 2s. How well does the rule do? *Right* = 1 if *class* = *group.*

```
CREATE      ; class = 2 - (x’d  >  .5 * (xbar1’d + xbar2’d))
            ; right = (class = group)  $
```

Examine the results.

```
CALC        ; hit rate = Xbr(right) $
LIST        ; group,class,right $
```

Is the correct classification rate significant?

```
CROSSTAB    ; Lhs = group ; Rhs = class $
```

Example 6. Hausman Test for Fixed vs. Random Effects

The Hausman (1978) test is used in the following setting: There are two estimators of the parameter vector $\boldsymbol{\beta}$, $\mathbf{b}_0$ and $\mathbf{b}_1$. Under H_0, $\mathbf{b}_0$ is consistent and efficient, but $\mathbf{b}_1$ is inconsistent. Under H_1, both estimators are consistent, but $\mathbf{b}_0$ is inefficient. The statistic is computed using

$$H = (\mathbf{b}_0 - \mathbf{b}_1)'[\text{Est.Asy.Var}(\mathbf{b}_0) - \text{Est.Asy.Var}(\mathbf{b}_1)]^{-1}(\mathbf{b}_0 - \mathbf{b}_1).$$

A common application of the test is to distinguish fixed vs. random effects in a linear regression model.

H_0: The fixed effects model is appropriate. The preferred estimator is least squares with dummy variables. This is $\mathbf{b}_0$.

H_1: The random effects model is appropriate. The preferred estimator is generalized least squares. This is $\mathbf{b}_1$.

```
SETPANEL    ; Group = … the identification variable ; Pds = the panel spec. $
NAMELIST    ; x = ... not including one $
CALC        ; k = Col(x) $
REGRESS     ; Lhs = y ; Rhs = x ; Panel ; Fixed Effects $
MATRIX      ; b0 = b(1 : k)          ? This extracts the first K elements.
            ; v0 = varb $      LS dummy variable estimator
REGRESS     ; Lhs = y ; Rhs = x ; Panel ; Random Effects $
MATRIX      ; b1 = b(1 : k)   ? 2 step GLS estimator
            ; v1 = varb(1:k,1:k)
            ; d = b0 - b1
            ; vd = v0 - v1
            ; List ; hausman = d' * Sinv(vd) * d
            ; pvalue = 1 - Chi(hausman, k) $
```

Note the following about this test:

1. The Hausman statistic is computed using the Cholesky inversion program for symmetric positive definite matrices. It occasionally occurs that the difference matrix is not positive definite (PD). If you use an ordinary inversion program to compute the inverse, you may get a misleading, or even negative result for the Hausman statistic, as the matrix may be nonsingular even if it is not positive definite. When the difference matrix is not PD, you should use zero for the Hausman statistic. Also, authors occasionally force the issue by using a generalized (G2) inverse for this computation. Once again, this can produce a misleading result. If the matrix is PD, the G2 inverse is not needed. If it is not PD, you should obtain 0.0 for the statistic, not the result of an ad hoc patch for the covariance matrix.

2. This statistic is computed and reported automatically by the panel data estimator if you do not specify either **; Fixed** or **; Random** in the **REGRESS** command. See the example below.

NOTE: There is now a built in function to compute this statistic. For the routine above, the full computation would be returned by **MATRIX ; Hsmn(bl,vl,b0,v0) $**

The following applies the routine to the Cornwell and Rupert labor supply data used in Greene (2012, Table F8.1). Some of the intermediate results are omitted. The last line redoes the computation with *LIMDEP*'s built-in routines.

```
NAMELIST    ; x = exp,wks,occ,ind,south,smsa,ms,union,one $
CALC        ; k = Col(x) - 1 $
REGRESS     ; Lhs = lwage ; Rhs = x ; Panel ; Fixed ; Pds = 7 $
MATRIX      ; bfe = b(1 : k)   ? This extracts the first K elements.
            ; vfe = varb $      LS dummy variable estimator
REGRESS     ; Lhs = lwage ; Rhs = x ; Panel ; Random ; Pds = 7 $
MATRIX      ; bre = b(1 : k)   ? 2 step GLS estimator
            ; vre = varb(1:k,1:k)
            ; d = bfe - bre
            ; vd = vfe - vre
            ; List ; hausman = d' * Sinv(vd) * d $
CALC        ; List ; pvalue = 1 - Chi(hausman, k) $
REGRESS     ; Lhs  = lwage ; Rhs = x ; Panel ; Pds = 7 $
```

```
--------------------------------------------------------------------------------
LSDV         least squares with fixed effects ....
LHS=LWAGE    Mean                 =       6.67635
             Standard deviation   =        .46151
             No. of observations  =          4165  Degrees of freedom
Regression   Sum of Squares       =       803.281         602
Residual     Sum of Squares       =       83.6239        3562
Total        Sum of Squares       =       886.905        4164
             Standard error of e  =        .15322
Fit          R-squared            =        .90571  R-bar squared =   .88978
Model test   F[602,  3562]        =      56.83746  Prob F > F*   =   .00000
Diagnostic   Log likelihood       =    2228.82754  Akaike I.C.   = -3.61859
             Restricted (b=0)     =   -2688.80603
Panel:Groups Empty      0,     Valid data      595
             Smallest   7,     Largest           7
             Average group size in panel      7.00
Variances    Effects a(i)        Residuals e(i,t)
              1.111677                    .023477
--------+-----------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
   LWAGE|  Coefficient       Error       z     |z|>Z*         Interval
--------+-----------------------------------------------------------------------
     EXP|     .09658***      .00119    81.10  .0000      .09424    .09891
     WKS|     .00114*        .00060     1.89  .0583     -.00004    .00232
     OCC|    -.02486*        .01389    -1.79  .0734     -.05208    .00236
     IND|     .02076         .01557     1.33  .1825     -.00976    .05127
   SOUTH|    -.00320         .03458     -.09  .9263     -.07096    .06457
    SMSA|    -.04373**       .01958    -2.23  .0256     -.08211   -.00534
      MS|    -.03026         .01914    -1.58  .1138     -.06777    .00725
   UNION|     .03416**       .01504     2.27  .0232      .00468    .06364
--------+-----------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
--------------------------------------------------------------------------------
Random Effects Model: v(i,t)    = e(i,t) + u(i)
Estimates:  Var[e]              =       .023477
            Var[u]              =       .128215
            Corr[v(i,t),v(i,s)] =       .845235
Lagrange Multiplier Test vs. Model (3) =3847.31
( 1 degrees of freedom, prob. value =  .000000)
(High values of LM favor FEM/REM over CR model)
Baltagi-Li form of LM Statistic =       3847.31
--------+-----------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
   LWAGE|  Coefficient       Error       z     |z|>Z*         Interval
--------+-----------------------------------------------------------------------
     EXP|     .05802***      .00090    64.32  .0000      .05625    .05979
     WKS|     .00163***      .00060     2.72  .0065      .00046    .00280
     OCC|    -.11293***      .01280    -8.82  .0000     -.13803   -.08784
     IND|    -.01361         .01405     -.97  .3326     -.04115    .01392
   SOUTH|    -.06737***      .02388    -2.82  .0048     -.11418   -.02057
    SMSA|    -.02141         .01668    -1.28  .1992     -.05410    .01128
      MS|    -.02516         .01729    -1.46  .1455     -.05905    .00872
   UNION|     .03968***      .01381     2.87  .0041      .01261    .06674
Constant|    5.55102***      .04166   133.23  .0000     5.46936   5.63269
--------+-----------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
--------------------------------------------------------------------------------
HAUSMAN |             1
--------+--------------
       1|       2704.39
[CALC] PVALUE  =       .0000000
```

This is reported by the last command above, which fits both models.

```
-----------------------------------------------------------------------------
Random Effects Model: v(i,t)    = e(i,t) + u(i)
Estimates:  Var[e]              =       .023477
            Var[u]              =       .128215
            Corr[v(i,t),v(i,s)] =       .845235
Lagrange Multiplier Test vs. Model (3) =3847.31
( 1 degrees of freedom, prob. value =  .000000)
(High values of LM favor FEM/REM over CR model)
Baltagi-Li form of LM Statistic =       3847.31
Moulton/Randolph form:SLM N[0,1]=         62.94
Fixed vs. Random Effects (Hausman)     =2704.39
( 8 degrees of freedom, prob. value =  .000000)
(High (low) values of H favor F.E.(R.E.) model).
            Sum of Squares          1977.160321
            R-squared                 -1.229281
--------+--------------------------------------------------------------------
```

Using the built in function for the Hausman test,

MATRIX ; Hsmn(bre,vre,bfe,vfe) $

we obtain the same result,

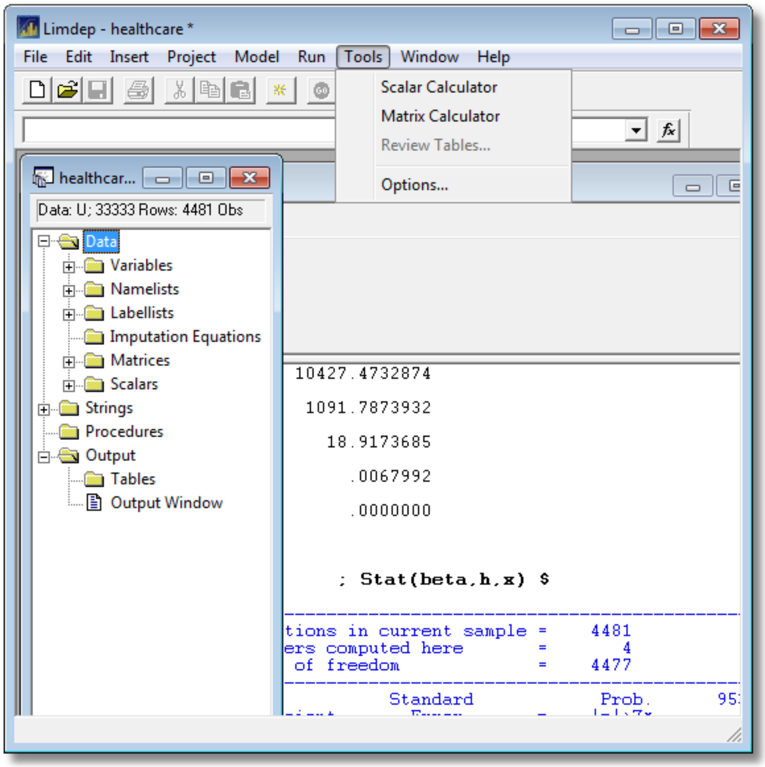
```
-------------------------------------------------
Hausman Test Statistic
                        Estimator      Covariance
Efficient   Estimator   BRE            VRE
Inefficient Estimator   BFE            VFE
Degrees of freedom  8   Chi squared    P value
                         2704.39       .0000
-------------------------------------------------
```

R16.2 Entering MATRIX Commands

MATRIX commands are typically given as parts of programs that perform larger functions, such as in the examples in Section R16.1. You also have a matrix ‘calculator’ that you can access occasionally in a window that is separate from your primary desktop windows (project, editing, and output).

R16.2.1 The Matrix Calculator

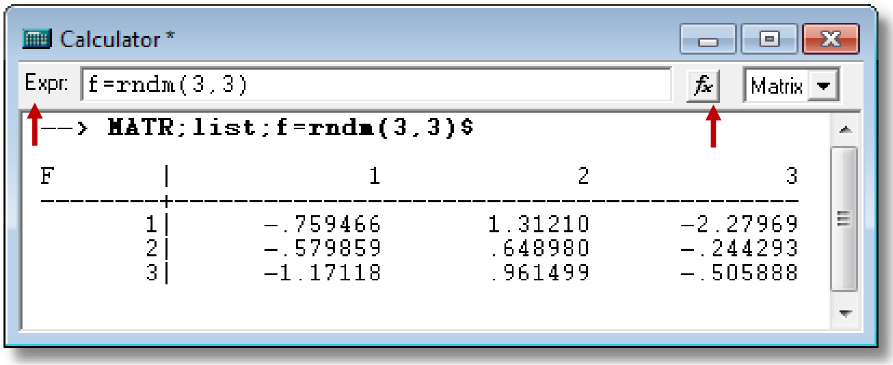
You may invoke the matrix calculator by selecting Tools:Matrix Calculator as shown in Figure R16.1.



```
10427.4732874

 1091.7873932

   18.9173685

     .0067992

     .0000000


   ; Stat(beta,h,x) $

-------------------------------------------
tions in current sample =    4481
ers computed here       =       4
 of freedom             =    4477
-------------------------------------------
           Standard             Prob.    95%
```

Figure R16.1 Tools Menu for Matrix Calculator

The matrix calculator window is shown in Figure R16.2.



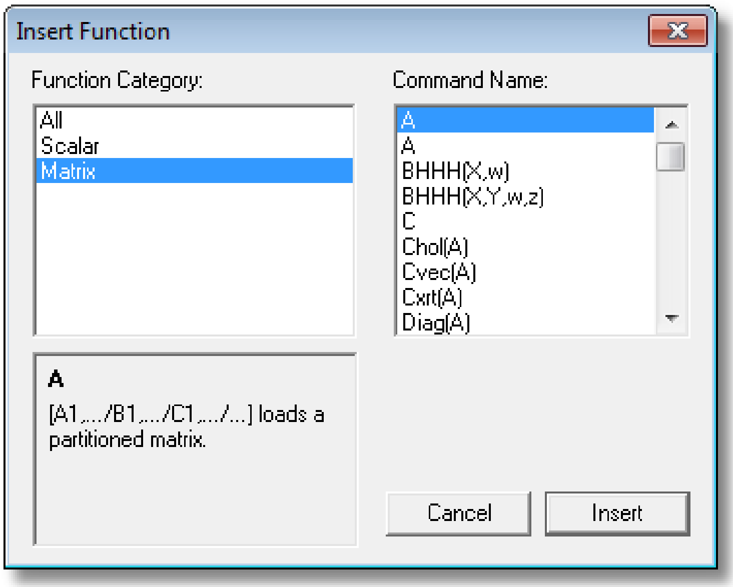
```
f=rndm(3,3)

--> MATR;list;f=rndm(3,3)$

F        |         1           2           3
---------+------------------------------------
        1|   -.759466     1.31210    -2.27969
        2|   -.579859     .648980    -.244293
        3|   -1.17118     .961499    -.505888
```

Figure R16.2 Matrix Calculator Window

You can leave the matrix calculator window open while you go to some other function. For example, you may find it convenient to interrupt your work in the editing/output windows by activating the matrix calculator to check some result which, perhaps, is not emerging the way you expected.

There are two other ways to enter commands in the matrix calculator window. You can type **MATRIX** commands in the smaller ‘Expr:’ (expression) window. In this dialog mode, if your command will not fit on one line, just keep typing. At some convenient point, the cursor will automatically drop down to the next line. Only press Enter when you are done entering the entire command. In this mode of entry, you do not have to end your commands with a **$**.

Alternatively, you can click the f_x button to open a subsidiary window that provides a menu of the functions (procedures). See Figure R16.3.



A

[A1,.../B1,.../C1,.../...] loads a partitioned matrix.

Figure R16.3 Insertion Window for Matrix Functions

You can select the function you wish to insert in your command. You must then fill in the arguments of the function that are specific to your expression. (E.g., if you want **Chol(sigma)**, you can select Chol(A) from the menu, then you must change ‘A’ to ‘sigma.’ in the command.)

R16.2.2 MATRIX Commands

If your **MATRIX** command is part of a program, it is more likely that you will enter it ‘in line,’ rather than in the matrix calculator. That is as a command in the text editor, in the format,

MATRIX ; ... the desired command ... $

Commands may be entered in this format from the editor, as part of a procedure, or in an input file. All of the applications given elsewhere in this manual are composed of in line commands, as are the examples given in Section R16.1 and in many places in the preceding chapters.

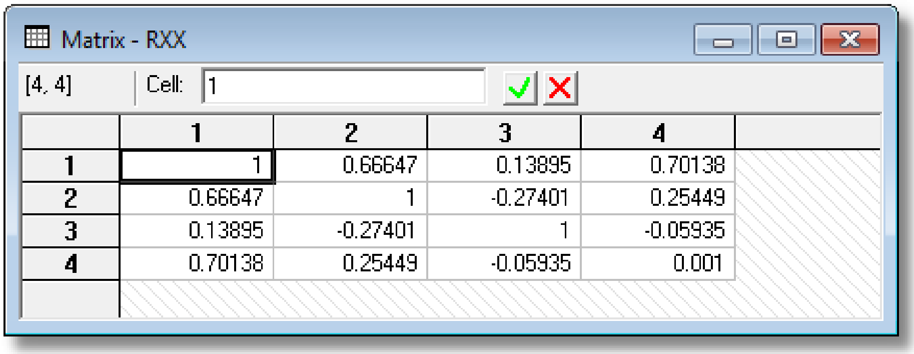
The essential format of a **MATRIX** command is

MATRIX ; name = result ; ... additional commands ... $

If you wish to see the ‘result’ but do not wish to keep it, you may omit the ‘**; name =**.’ For example, you are computing a result and you receive an unexpected diagnostic. We sometimes come across a matrix, say *rxx*, that we thought was positive definite, but when we try something like **MATRIX ; Sinv(rxx) $**, a surprise error message that the matrix is not positive definite shows up. A simple listing of the matrix shows the problem. The .001 in the 4,4 element is supposed to be a 1.0. Now we have to go back and find out how the bad value got there – some previous calculation did something unexpected.

```
MATRIX ; Sinv(Rxx) $
Error   185: MATRIX - GINV,SINV,CHOL  singular, not P.D. if SINV or CHOL
MATRIX ; List ; Rxx $
Result  |           1              2              3              4
--------+----------------------------------------------------------
       1|     1.00000        .666470        .138950        .701380
       2|     .666470        1.00000       -.274010        .254490
       3|     .138950       -.274010        1.00000      -.0593500
       4|     .701380        .254490      -.0593500      .00100000
```

If you want only to see a matrix, and not operate on it, you can just double click its name in the project window. That will open a window that displays the matrix. The offending *rxx* matrix shown above is displayed in Figure R16.4.



	1	2	3	4
1	1	0.66647	0.13895	0.70138
2	0.66647	1	-0.27401	0.25449
3	0.13895	-0.27401	1	-0.05935
4	0.70138	0.25449	-0.05935	0.001

Figure R16.4 Matrix Display from the Project Window

Matrix results will be mixtures of matrix algebra, i.e., addition, multiplication, subtraction, and matrix functions, such as inverses, characteristic roots, and so on, and, possibly, algebraic manipulation of functions of matrices, such as products of inverses.

R16.2.3 Conditional Commands

All **MATRIX** commands may be made conditional, in the same manner as **CREATE** or **CALCULATE**. The conditional command would normally appear

MATRIX ; If (logical expression) name = expression $

The logical expression may be any expression that resolves either to 'true' or 'false' or to a numeric value, with nonzero implying true. The rules for the expression are identical to those for **CREATE** (see Section R4.2.2) and **REJECT** (see Section R7.2.2), as well as **CALCULATE**, and all forms of **DO**. In this setting, if the condition is true, '*name*' is computed; if it is false, '*name*' is not computed. Thus, if *name* is a new matrix, and the condition is false, after the command is given, *name* will not exist. For example,

MATRIX ; If (a(1,1) > rsqrd) q = Dtrm(v) $

An entire set of **MATRIX** commands can be made conditional by placing a semicolon after the condition, as in

MATRIX ; If (condition) ; name = result ; result $

If the condition is false, none of the commands that follow it are carried out. This form of condition may appear anywhere in a group of **MATRIX** commands.

You may also make an entire set of matrix calculations conditional with the syntax

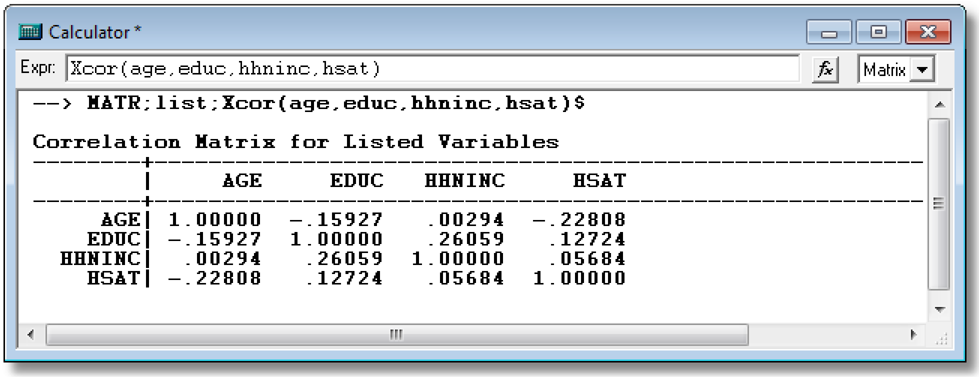
MATRIX ; If (condition) | a set of matrix results $

If the condition is false, none of the commands after the bar are carried out. The test statistic for the Brant test in Section R16.4.5 provides an extensive example.

R16.3 Matrix Output

The results of **MATRIX** commands can be matrices with up to 50,000 elements, and can thus produce enormous amounts of output. As such, most of the display of matrix results is left up to your control.

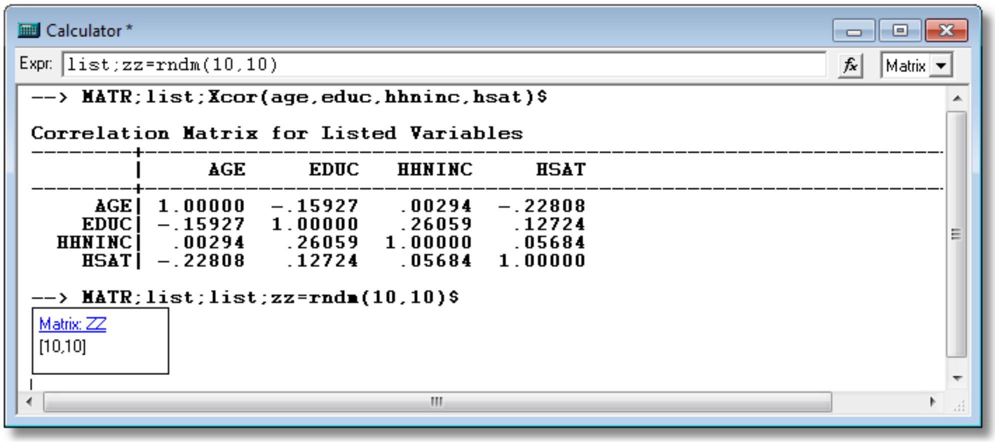
Matrix results are always displayed in the calculator window. When commands are in line, results are generally not shown unless you specifically request the display with **; List**. (See Section R16.3.1.) The figures below demonstrate.



```
Xcor(age,educ,hhninc,hsat)

--> MATR;list;Xcor(age,educ,hhninc,hsat)$

Correlation Matrix for Listed Variables
--------+-----------------------------------------------------
        |      AGE      EDUC    HHNINC      HSAT
--------+-----------------------------------------------------
     AGE|  1.00000  -.15927    .00294  -.22808
    EDUC| -.15927   1.00000    .26059   .12724
  HHNINC|   .00294   .26059   1.00000   .05684
    HSAT| -.22808    .12724    .05684  1.00000
```

Figure R16.5 Matrix Result in the Calculator Window

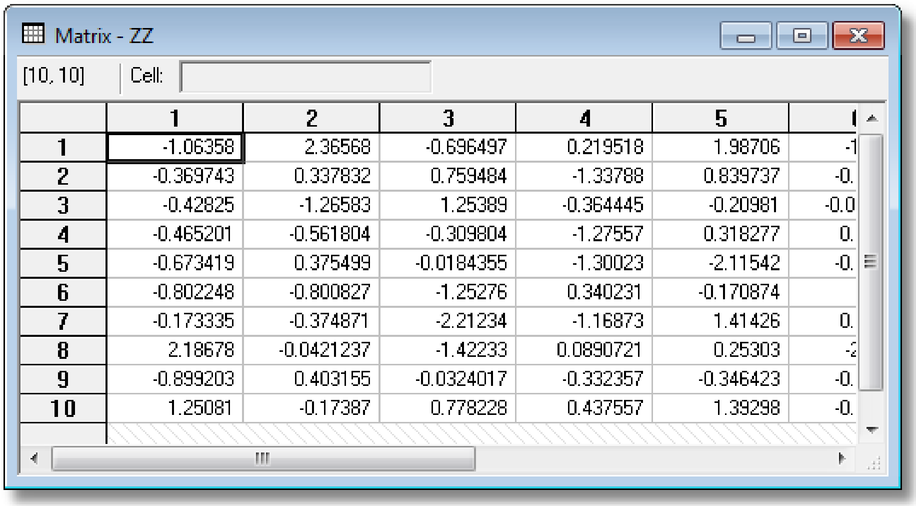
When the computed result has more than five columns or more than 20 rows, it will be shown in the output window as a place holder (object).



```
list;zz=rndm(10,10)

--> MATR;list;Xcor(age,educ,hhninc,hsat)$

Correlation Matrix for Listed Variables
--------+-----------------------------------------------------
        |      AGE      EDUC    HHNINC      HSAT
--------+-----------------------------------------------------
     AGE|  1.00000  -.15927    .00294  -.22808
    EDUC| -.15927   1.00000    .26059   .12724
  HHNINC|   .00294   .26059   1.00000   .05684
    HSAT| -.22808    .12724    .05684  1.00000

--> MATR;list;list;zz=rndm(10,10)$
Matrix: ZZ
[10,10]
```

Figure R16.6 Matrix Result in the Output Window

If you double click the object, you can display the full matrix in a scalable window.



Matrix - ZZ

[10, 10] Cell:

	1	2	3	4	5	
1	-1.06358	2.36568	-0.696497	0.219518	1.98706	-1
2	-0.369743	0.337832	0.759484	-1.33788	0.839737	-0.
3	-0.42825	-1.26583	1.25389	-0.364445	-0.20981	-0.0
4	-0.465201	-0.561804	-0.309804	-1.27557	0.318277	0.
5	-0.673419	0.375499	-0.0184355	-1.30023	-2.11542	-0.
6	-0.802248	-0.800827	-1.25276	0.340231	-0.170874	
7	-0.173335	-0.374871	-2.21234	-1.16873	1.41426	0.
8	2.18678	-0.0421237	-1.42233	0.0890721	0.25303	-2
9	-0.899203	0.403155	-0.0324017	-0.332357	-0.346423	-0.
10	1.25081	-0.17387	0.778228	0.437557	1.39298	-0.

Figure R16.7 Matrix Display Window

You can navigate around any matrix in the display with the editing keys, arrow, PgUp, and so on.

TIP: If you double click the upper left (blank) box in the window, this will 'select' the entire window. You can then use edit copy/paste to bring this (tab delimited) matrix directly into *Excel*.

R16.3.1 Matrix Results

When **MATRIX** commands are given in line, the default is not to display the results of any matrix computations on the screen or in the output file. It is assumed that in this mode, results are mostly intermediate computations. The output file will contain, instead, a listing of the matrix expression and either a confirmation that the result was obtained or just a statement of the expression with a diagnostic in the trace file. For example, the command

MATRIX ; a = Iden(20) $

produces only an echo of the expression.

You can request full display of matrices in the output file by placing

; List

before the matrix to be listed. Note how this has been used extensively in the preceding examples in Section R16.1. This is a switch that will now remain on until you turn it off with **; Nolist**. When the end of a command is reached, **; Nolist** once again becomes the default. The **; Nolist** and **; List** switches may be used to suppress and restore output at any point. When the **; Nolist** specification appears in a **MATRIX** command, no further output appears until the **; List** specification is used. At the beginning of a command, the **; List** switch is off, regardless of where it was before. If you are doing many computations, you can suppress some of them, then turn the output switch back on, in the middle of a command. For example:

MATRIX ; Nolist ; xxi = <x1'x1> ; List ; Root(xxi) $

displays only the characteristic roots of the inverse of a particular **X′X** matrix. Neither *x*1′*x*1 nor *xxi* are displayed.

Displaying matrices that already exist in the matrix work area requires only that you give the names of the matrices. I.e.,

MATRIX ; List ; abcd ; qed $ (note, separated by semicolons, not commas)

would request that the matrices named *abcd* and *qed* be displayed on your screen. You might also want to see the results of a matrix procedure displayed, without retaining the results. The following are some commands that you might type:

MATRIX	**; Root(xx) $**	lists characteristic roots of *xx*.
MATRIX	**; a* b $**	displays the matrix product *ab*.
MATRIX	**; Mean(x*) $**	displays the means of all variables whose names begin with *x*.

These commands just display the results of the computations; they do not retain any new results.

R16.3.2 Unformatted Output

In the cases considered thus far, when a matrix is listed in your output, it is partitioned and formatted for convenient viewing. However, this may make further analysis of the matrix inconvenient. If you would like to produce an unmodified, unformatted copy of one or more matrices in your output so that you can manipulate them later, for example, use them in some other program, use the command

WRITE ; ... desired list of matrices $

The example below shows the difference in the two types of listings:

```
MATRIX ; List ; zz = Rndm(6,5)$
ZZ      |            1              2              3              4              5
--------+----------------------------------------------------------------------
       1|      -1.44814        1.36184       -.468357       -.622546        .795314
       2|      -.168379        .104536        .904864     -.00397597        .528702
       3|      .0952065        1.04398      -.0338957       -.769662       -1.44197
       4|      .0849252        .243405       -.673963        1.29946        1.20323
       5|      -.651083        .313592        .999907       .0339046        .104310
       6|       .679254       -1.55387        .562071       -2.44212        1.35792

WRITE ; zz $
[    6 by     5] Matrix ; zz       =[
      -1.448137,       1.361837,      -.4683571,      -.6225460,       .7953138/
      -.1683794,       .1045356,       .9048638,  -.3975969E-02,       .5287020/
   .9520647E-01,       1.043977,  -.3389574E-01,      -.7696619,      -1.441973/
   .8492519E-01,       .2434048,      -.6739631,       1.299464,       1.203225/
      -.6510833,       .3135922,       .9999074,   .3390462E-01,       .1043099/
       .6792535,      -1.553875,       .5620706,      -2.442124,       1.357922
]$
```

The matrix is displayed as a command, but the body of the matrix is comma and slash delimited.

R16.3.3 Technical Output

All computations in *LIMDEP* are done in 'double precision.' That means that although the visible displays of results typically contain anywhere from five to eight significant digits, all internal results are computed with 17 significant digits. For some purposes, for example, for checking the accuracy of iterative programs that you write, you may wish to see all of the computed digits for a matrix result. You can request this format by using

; Peek

in your **MATRIX** command at the point at which you wish to begin the technical display. The listing below shows the internal form of the first several values in *zz* from above.

```
Display of all internal digits of matrix Result
Result[1,1]=-.14481369392191410D+01
Result[1,2]=.13618369140362890D+01
Result[1,3]=-.46835710105771710D+00
Result[1,4]=-.62254596617409810D+00
Result[1,5]=.79531377257669170D+00
Result[2,1]=-.16837941022982930D+00
Result[2,2]=.10453556133910670D+00
Result[2,3]=.90486384151079990D+00
Result[2,4]=-.39759692924754610D-02
Result[2,5]=.52870199710107050D+00
Result[3,1]=.95206471103564530D-01
Result[3,2]=.10439772152053720D+01
Result[3,3]=-.33895735151328110D-01
Result[3,4]=-.76966187735011250D+00
...
```

R16.3.4 Exporting Matrix Results from *LIMDEP*

You can export your statistical results to other packages. In Section R16.3, we noted that with edit/copy and edit/paste, you can extract matrix results and drop them directly into spreadsheet programs. You can also export your results more formally to any program that can accept the 'comma separated values,' or CSV format, such as *Excel*. The file that *LIMDEP* creates can be read directly, without any further manipulation on your part. Setting it up requires a few steps, as shown below.

Step 1. Open the file that will contain the results to be exported.

This will be a CSV (comma separated values) file. Use the following **OPEN** command:

OPEN ; Export = …<filename>.csv $

You must open the file with extension .csv for this operation to succeed. *LIMDEP* does not check this file setup for you – the program assumes that the file is opened correctly.

Step 2. Use the ; Export specification in your model commands.

In specific model commands that you wish to export, use the model option **; Export** to put a table of coefficients, etc. in the export file. You may also use **; Title = up to 80 characters** to put a line of text at the top of the results. For some other specific commands, you can use

MATRIX	**; Export = list of matrices $**	puts a list of matrices in the file.
DSTAT	**; Export ; Rhs = ... $**	copies the results to the CSV file.
CALC	**; Export = list of scalars $**	copies scalars to the file.

Step 3. Close the file before you try to use it.

When you are finished exporting results to the file, use

```
CLOSE      ; Export $
```

to end accumulation of results in the file.

After this file is created, you can now export your results to *Excel* just by double clicking the file name in any context, such as Windows Explorer. There are two possible conflicts to be wary of:

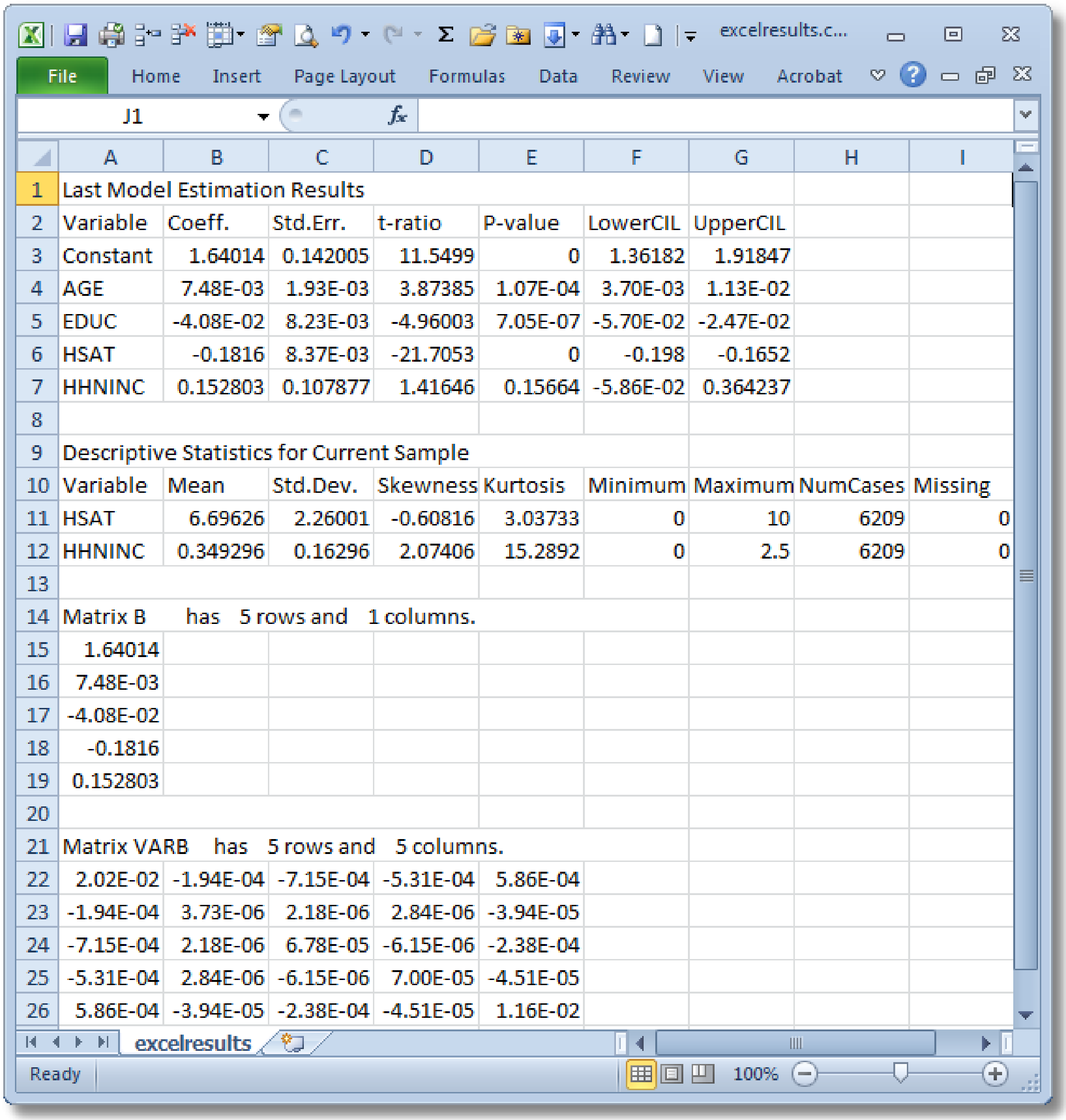
- The file cannot be reopened. If you repeat an **OPEN ; Export = name $** command, the original file is erased and a new one with that name is created.

- Do not use this file, e.g., by *Excel*, until you exit *LIMDEP*, even if you have used a **CLOSE** command to close the file.

An example follows: We first create the file in *LIMDEP*.

```
OPEN       ; Export = “C:\work\excelresults.csv” $
PROBIT     ; Export ; Lhs = doctor
           ; Rhs = one,age,educ,hsat,hhninc $
DSTAT      ; Rhs = hsat,hhninc $
MATRIX     ; Export = b,varb $
```

We then open the file in *Excel* with the results shown in Figure R16.8.



	A	B	C	D	E	F	G	H	I
1	Last Model Estimation Results								
2	Variable	Coeff.	Std.Err.	t-ratio	P-value	LowerCIL	UpperCIL		
3	Constant	1.64014	0.142005	11.5499	0	1.36182	1.91847		
4	AGE	7.48E-03	1.93E-03	3.87385	1.07E-04	3.70E-03	1.13E-02		
5	EDUC	-4.08E-02	8.23E-03	-4.96003	7.05E-07	-5.70E-02	-2.47E-02		
6	HSAT	-0.1816	8.37E-03	-21.7053	0	-0.198	-0.1652		
7	HHNINC	0.152803	0.107877	1.41646	0.15664	-5.86E-02	0.364237		
8									
9	Descriptive Statistics for Current Sample								
10	Variable	Mean	Std.Dev.	Skewness	Kurtosis	Minimum	Maximum	NumCases	Missing
11	HSAT	6.69626	2.26001	-0.60816	3.03733	0	10	6209	0
12	HHNINC	0.349296	0.16296	2.07406	15.2892	0	2.5	6209	0
13									
14	Matrix B	has	5 rows and	1 columns.					
15	1.64014								
16	7.48E-03								
17	-4.08E-02								
18	-0.1816								
19	0.152803								
20									
21	Matrix VARB	has	5 rows and	5 columns.					
22	2.02E-02	-1.94E-04	-7.15E-04	-5.31E-04	5.86E-04				
23	-1.94E-04	3.73E-06	2.18E-06	2.84E-06	-3.94E-05				
24	-7.15E-04	2.18E-06	6.78E-05	-6.15E-06	-2.38E-04				
25	-5.31E-04	2.84E-06	-6.15E-06	7.00E-05	-4.51E-05				
26	5.86E-04	-3.94E-05	-2.38E-04	-4.51E-05	1.16E-02				

Figure R16.8 Results Exported to *Excel*

As noted earlier, you can also use edit/copy in a matrix window and edit/paste in your spreadsheet program to move matrices to other software.

R16.3.5 Matrix Statistical Output

Your matrix procedures will often create coefficient vectors and estimated covariance matrices for them. For any vector, *beta*, and square matrix, *v*, of the same order as *beta*, the command

MATRIX ; Stat(beta,v) $

will produce a table which assumes that these are a set of statistical results. The table contains the elements of *beta*, the diagonal elements of *v* and the ratios of the elements of *beta* to the square root of the corresponding diagonal element of *v* (assuming it is positive). For example, the listing below shows how the Stat function would redisplay the model results produced by a **LOGIT** command.

NOTE: MATRIX ; Stat(vector,matrix) $ has no way of knowing that the matrix you provide really is a covariance matrix or that it is the right one for the vector that precedes it. It requires only that '*vector*' be a vector and '*matrix*' be a square matrix of the same order as the vector. You must insure that the parts of the command are appropriate.

The routine to produce model output for a matrix computed set of results can be requested to display variable names by adding a namelist with the appropriate variables as a third argument in the **MATRIX ; Stat(b,v) $** function. If your estimator is a set of parameters associated with a set of variables, *x*, they are normally labeled *b*_1, *b*_2, etc. Adding the namelist to the **MATRIX ; Stat(b,v,x) $** function carries the variable labels into the function. An example follows: (Some results are omitted.)

Results from logit regression:

```
-----------------------------------------------------------------------------
Binary Logit Model for Binary Choice
Dependent variable                   MODE
Log likelihood function      -419.59809
Restricted log likelihood    -472.36152
Chi squared [    4 d.f.]      105.52685
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
    MODE|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|     .29245         .22832     1.28  .2002     -.15504     .73995
      GC|     .02160***      .00751     2.88  .0040      .00688     .03632
    TTME|    -.04387***      .00486    -9.03  .0000     -.05340    -.03435
    INVC|    -.00363         .00760     -.48  .6329     -.01852     .01126
    INVT|    -.00466***      .00106    -4.37  .0000     -.00674    -.00257
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

MATRIX ; Stat(b,varb) $

```
-----------------------------------------------------------------------------
Number of observations in current sample =     840
Number of parameters computed here       =       5
Number of degrees of freedom             =     835
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  Matrix|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     B_1|     .29245         .22832     1.28  .2002     -.15504    .73995
     B_2|     .02160***      .00751     2.88  .0040      .00688    .03632
     B_3|    -.04387***      .00486    -9.03  .0000     -.05340   -.03435
     B_4|    -.00363         .00760     -.48  .6329     -.01852    .01126
     B_5|    -.00466***      .00106    -4.37  .0000     -.00674   -.00257
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

MATRIX ; Stat(b,varb,x) $

```
-----------------------------------------------------------------------------
Number of observations in current sample =     840
Number of parameters computed here       =       5
Number of degrees of freedom             =     835
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  Matrix|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .29245         .22832     1.28  .2002     -.15504    .73995
      GC|     .02160***      .00751     2.88  .0040      .00688    .03632
    TTME|    -.04387***      .00486    -9.03  .0000     -.05340   -.03435
    INVC|    -.00363         .00760     -.48  .6329     -.01852    .01126
    INVT|    -.00466***      .00106    -4.37  .0000     -.00674   -.00257
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The **DISPLAY** command can be used to produce a similar set of results.

DISPLAY ; Parameters = b; Covariance = varb ; Labels = x
; Title = Logit Model for Mode Choice $

```
-----------------------------------------------------------------------------
Logit Model for Mode Choice
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  Matrix|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .29245         .22832     1.28  .2002     -.15504    .73995
      GC|     .02160***      .00751     2.88  .0040      .00688    .03632
    TTME|    -.04387***      .00486    -9.03  .0000     -.05340   -.03435
    INVC|    -.00363         .00760     -.48  .6329     -.01852    .01126
    INVT|    -.00466***      .00106    -4.37  .0000     -.00674   -.00257
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

R16.3.6 Descriptive Statistics for the Elements in a Matrix

The matrix function Dsta computes the mean and standard deviation of the elements in a matrix. The matrix may be any size or shape. Use

MATRIX ; Dsta(list of matrices) $

The matrices in the list need not be the same size. The following computes means and standard deviations for 5×5 and 10×10 matrices of random draws from the standard normal distribution.

MATRIX ; a5 = Rndm(5,5) ; a10 = Rndm(10,10) $
MATRIX ; Dsta(a5,a10) $

```
DESCRIPTIVE STATISTICS FOR MATRIX ELEMENTS
Matrix         Mean       Standard Deviation     Rows     Columns      Elements
A5          -.13648D+00       .94047D+00           5           5           25
A10          .55930D-01       .99133D+00          10          10          100
```

R16.3.7 Plotting Matrices

The elements of one matrix may be plotted against those of another with

MPLOT ; Rhs = matrix1 ; Lhs = matrix2 ; ... $

The rest of the command is the same as that for **PLOT** using variables. The two matrices must have the same dimensions, but they need not be vectors. For this command, the figure drawn is a scatter plot of the elements of *matrix*1 against the corresponding elements of *matrix*2. The **PLOT** command and options for **PLOT** and **MPLOT** are discussed in Chapter E6.

NOTE: *This is not a matrix function;* **MPLOT** *is a separate LIMDEP command.* Example 3 in Section R16.1 contains an application which uses **MPLOT**.

R16.4 Matrix Work Areas

You have a workspace for the results of matrix computations that contains roughly 500,000 cells plus another 100,000 for intermediate results of your estimation commands. Most of your matrix computations will take place in this area, either to manipulate matrices already in it or to use the data to compute matrices to add to it. You can define up to 100 named matrices. The maximum size of the result of a matrix computation is 50,000 cells. Although these dimensions may seem limited, because of the way matrices are defined by *LIMDEP*, it is unlikely that they will ever constrain you.

If necessary, you can delete matrices to make room in your workspace with

MATRIX ; Delete name, name, … $

(Note that there is no equals sign or colon after **; Delete**.) You can also delete matrices in the project window simply by selecting the matrix by name in the project window and pressing the Del key.

R16.4.1 Rebuilding the Matrix Work Area

When you open a project file, you restore your data and results, including the matrix work area. A side effect of this is that if you wish to restore just your matrix work area, you must also replace your entire data analysis project. The **MATRIX** functions Mput and Mget are provided to allow you to save and restore a matrix algebra 'subsession.' Mput creates a special file that contains the names, dimensions, and contents of all matrices that exist at the moment when it is created. Mget will read that file and restore the matrix work areas to their previous state, without changing anything else in your project. The syntaxes of these commands are

```
MATRIX      ; Mput = filename $
```

saves the matrix part of a **SAVE** (.lpj) file in the indicated file. The following command

```
MATRIX      ; Mget = filename $
```

restores a matrix algebra workspace while leaving the rest of the project unchanged. Recognizing the ambiguity of the location of a file in Windows, remember that you can use Insert:File Path to locate a matrix file. Also, though these look like project files, they have several layers of internal protection which will prevent them from accidentally overwriting the rest of your project.

You can clear all matrices out of the matrix work area with the command

```
MATRIX      ; Reset $
```

R16.4.2 Naming and Notational Conventions

Every numeric entity in *LIMDEP* is a matrix, and you will rarely have to make a distinction among them. For example, in the expression,

```
MATRIX      ; f = q ' r $
```

q and *r* could be any mix of:

- variables,
- data matrices,
- computed matrices,
- named scalars,
- literal numbers, e.g., 2.345,
- the number (symbol) 1, which has special meaning in matrix multiplication.

The two entities must be conformable for the matrix multiplication, but there is no requirement that two matrices be the same type of entity. (Usually, they will be.) For convenience, we will sometimes make the following definitions:

• Variable names are	*vnames.*
• Namelists are	*xnames.*
• Computed matrices are	*mnames.*
• Scalars are	*rnames.*
• Numbers are	*scalars.*

At any time, you can examine the contents of the tables of these names in your project workspace, just by clicking the particular name in your project window. Whenever you create an entity in any of these tables, all of the others are checked for conflicts. For example, if you try to create a variable named *q*, and there is already a matrix with that name, an error will occur.

NOTE: There are two reserved matrix names in *LIMDEP*. The matrix program reserves the names *b* and *varb* for the results of estimation programs. These two names may not appear on the left hand side of a matrix expression. They may appear on the right, however.

There are a few additional names which are read only some of the time. For example, after you use the **SURE** command, *sigma* becomes a reserved name. Model output will indicate if a reserved name has been created.

In the descriptions of matrix operations to follow,

- *xname* is the name of a data matrix. This will usually be a namelist. However, most data manipulation commands allow you merely to give a set of variable names instead.
- *mname* is the name of a computed matrix.
- *s* is a scalar. It may be a number or the name of a scalar which takes a value.
- A matrix has *r* rows and *c* columns.
- Matrices in matrix expressions are indicated with boldfaced uppercase letters.
- The transpose of matrix in a matrix algebra expression **C** is denoted **C** ′.
- The apostrophe, ’, also indicates transposition of a matrix in *LIMDEP* commands.
- The ordinary inverse of matrix **C** is denoted $\mathbf{C}^{-1}$.

The result of a procedure that computes a matrix **A** is denoted *a*. Input matrices are *c*, *d*, etc. In any procedure, if *a* already exists, it may appear on both sides of the equals sign with no danger of ambiguity; all matrices are copied into internal work areas before the operation actually takes place. Thus, for example, a command may replace *a* with its own transpose, inverse or determinant. You can replace a matrix with some function of that matrix which has different dimensions entirely. For example, you might replace the matrix named *a* with *a*′*a* or with *a*’s rank, trace or determinant.

Note in these definitions and in all that follows, we will make a distinction between a matrix expression (in theory), such as $\mathbf{F} = (1/n)\mathbf{X}'\mathbf{X}$, and the entities that you manipulate with your *LIMDEP* commands, for example, ‘you have created *f* = *x*’*x*.’ There are thus three sets of symbols. We will use bold upper case symbols in matrix algebra descriptions; we will use bold lower case symbols for the parts of *LIMDEP* commands. We will use italic lower case symbols when we refer, outside *LIMDEP* commands, to the names of matrices, variables, namelists and scalars you have created. Consider, for example, the following: ‘The sample second moment matrix of the data matrix **X** is **F** = (1/n)**X′X**. You can compute this by defining **X** with a command such as **NAMELIST ; x = one,age,income $**, then using the command **MATRIX ; f = 1/n*x’x $**. After you execute this command, you will see the matrix *f* in your project window listing of matrices. The namelist *x* will also appear in the project window list of namelists.’ You might note, we have used this convention at several points above.

R16.4.3 Matrix Dimensions

The dimensions of all matrices are stored and kept automatically and almost never have to be given explicitly. A 1×1 matrix is usually not treated any differently from any other matrix when it is the result of a procedure. For example,

```
MATRIX        ; detc = Dtrm(c) $
```

computes a 1×1 matrix which equals the determinant of **C**.

A row vector is rarely the same as a column vector. *LIMDEP* will not let you add a 3×1 vector to a 1×3 vector. With only a few exceptions that are made explicit below, all matrices used in all computations must be strictly conformable.

R16.4.4 Placing Matrix Results in Scalars

Matrix operations may specify that the result should be placed in an existing scalar instead. (See Chapter R17.) If the result is a 1×1 matrix, the result is placed in the indicated scalar. If the result is a more general matrix, the (1,1) element of the result is placed in the scalar and the remaining elements are lost. For example, suppose *x* is a five column data matrix. That is, *x* would be defined with a **NAMELIST** command. Then

```
CALC          ; varx1 = 0 $ Must already exist to use as a matrix result
MATRIX        ; cov = Xvcm(x)
              ; varx1 = Xvcm(x) $
```

creates *cov*, a 5×5 covariance matrix, and places the variance of the first variable in *varx*1.

You can also use an element of a matrix in any later computation just as if it were a scalar. Consider the following example: The **MATRIX** command computes a *K*×*K***X′X** matrix, then turns it into a 1×1 matrix equal to its own determinant. The **CALC** command then computes the log of this determinant by computing the log of the first (and only) element in *xx*.

```
MATRIX        ; xx = Dtrm(x'x) $
CALC          ; logdet = Log(xx) $
```

TIP: There are 86 user defined scalars available to you. But, in view of the preceding, you can create up to 100 more by using 1×1 matrices. We will return to this issue in Chapter R17.

Note, as well, there are many functions that are common in econometrics provided to simplify computations such as this. For example, the computation above is obtained with the simple command,

```
MATRIX        ; logdet = Logd(x'x) $.
```

R16.4.5 Compound Names for Matrices, Variables and Scalars

The names of matrices, variables and scalars may all be of the form *aaaa*:*ssss* where *ssss* is the name of a scalar. The scalar must take an integer value from 00 to 99. The value is appended to the name to make a variable with the compound name. This feature will be useful for looping in procedures. For example:

```
CALC          ; index = 1 $
PROCEDURE $
CREATE        ; x : index = 1 / index $
ENDPROCEDURE $
EXECUTE       ; index = 1,10 $
```

creates 10 variables, *x*1 = 1, *x*2 = 1/2, *x*3 = 1/3, *x*4 = 1/4, ..., *x*10 = 1/10. The Brant test for homogeneity in an ordered logit model provides another example – in this program, both matrices and variables are being given compound names. This extensive routine also illustrates many of the matrix computations discussed in this chapter. The discussion surrounding the commands will show the usages.

This is the Brant test for preference heterogeneity in an ordered logit model. The base model has Prob[*y*=*j*]=*F*[*m*(*j*)-**β′x**] - *F*[*m*(*j*-1)-**β′x**] The test examines whether **β** is the same for all outcomes. It is a Wald test of the hypothesis **β**(0)=**β**(1)=... when **β** is allowed to vary across choices. Each **β**(*j*) is estimated by the binary choice model Prob[*y* >*r*] for *r* = 0,1,...,Max(*y*)-1. For *y* = 0,1,2,3,4,5, there are five **β** vectors estimated, and the test then evaluates the *J* = 4 vector equalities. This routine is completely self contained. It requires only that the *x* and *y* be set up at the beginning. This procedure is limited to *y* taking values up to five. The pattern below shows how it could be extended if necessary. Only changes to the **CREATE** and **MATRIX** commands are needed to allow for more outcomes in the ordered choice model.

These commands set up the Lhs variable *y* and Rhs namelist *x*. Here, we are generating artificial data.

```
SAMPLE        ; 1-1000 $
CALC          ; Ran(12345) $
CREATE        ; y = Rnd(6) - 1 ; xa = Rnn(0,1) ; xb = Rnn(0,1) ; xc = Rnn(0,1) $
NAMELIST      ; x = xa,xb,xc $  x does not include a constant term.
```

The remainder of the program is generic and need not be changed by the user. These commands compute some values and matrix templates that are used later in the program. Matrices *i* and *mi* are an identity and negative of identity, *z* is a zero matrix; *bt* and *d* are empty here, and will be filled during execution of the procedure.

```
NAMELIST      ; x1 = x,one $
CALC          ; k = Col(x) $
CALC          ; ymax = Max(y ); y1  = ymax-1 ; kj = ymax*k ; k1j = y1*k $
MATRIX        ; i = Iden(k) ; z = Init(k,k,0) ; mi = -1*i $
MATRIX        ; bt = Init(kj,1,0) ; d = Init(k1j,kj,0) $
```

This procedure computes the individual logit equations. To reduce the number of commands, it makes heavy use of compound names. Loop index *y*1 takes values 1,2,...,*ymax*. *j* = *y*1 - 1, 0,1,2,...*ymax*-1. The procedure is creating variables *z*0, *z*1, ... each equal to a binary variable that equals one when *y*>*j*. It is creating coefficient vectors *b*0, *b*1, ... then injecting (stacking) them in the large vector *bt*. Each **LOGIT** command creates a variable with fitted probabilities, *p*0,... After each *b*:*j* is computed, a vector of derivatives, *w*0 = *p*0(1-*p*0), *w*1 = *p*1(1-*p*1),... is computed.

We are creating matrices *v*0, *v*1,... as inverses of moment matrices. Finally, the large matrix *d* is a partitioned matrix in which block row *j* contains *i* on the diagonal and -*i* at the end of the row.

```
PROC = Logits $
CALC        ; j = y1-1 ; jy = j*k+1 ; jyk = jy+k $
CREATE      ; z:j = y > j $
LOGIT       ; Lhs = z:j ; Rhs = x1 ; Prob = p:j $
MATRIX      ; b:j = b(1:k) ; bt(jy) = b:j $
CREATE      ; w:j = p:j*(1-p:j) $
CALC        ; jy = Min(jy,((ymax-2)*k+1)) ; jyk = jy+k $
MATRIX      ; v:j = <x1'[w:j]x1> ; vt = v:j ; vt = Part(vt,1,k,1,k) ; v:j = vt $
MATRIX      ; d(jy,1) = i ; d(jy,jyk) = mi $
ENDPROC $
EXECUTE     ; y1    = 1,ymax ; Silent $
```

Conditional **CREATE** commands compute derivatives for estimated models. These use the probabilities computed by the **LOGIT** commands in the procedure. The commands are conditional. They only compute the variables needed, depending on the number of outcomes in the ordered choice model

```
CREATE      ; If[j >= 1] | w01=p1-p0*p1  $
CREATE      ; If[j >= 2] | w02=p2-p0*p2 ; w12=p2-p1*p2 $
CREATE      ; If[j >= 3] | w03=p3-p0*p3 ; w13=p3-p1*p3 ; w23=p3-p2*p3 $
CREATE      ; If[j >= 4] | w04=p4-p0*p4 ; w14=p4-p1*p4
            ;              w24=p4-p2*p4 ; w34=p4-p3*p4 $
```

These are the partitioned covariance matrices. **V** is a partitioned matrix. The number of blocks depends on the number of outcomes in the choice model.

```
MATRIX      ; If(j >= 1) | v01=x1'[w01]x1 ; v01=v01(1:k,1:k)
            ; v01=v0*v01*v1; v10=v01' $
MATRIX      ; If(j >= 2) | v02=x1'[w02]x1 ; v02=v02(1:k,1:k)
            ; v02=v0*v02*v2; v20=v02'
            ; v12=x1'[w12]x1 ; v12=v1*v12*v2 ; v12=v12(1:k,1:k) ; v21=v12' $
MATRIX      ; If(j >= 3) | v03=x1'[w03]x1 ; v03 = v03(1:k,1:k)
            ; v03=v0*v03*v3 ; v30=v03'
            ; v13=x1'[w13]x1; v13=v13(1:k,1:k) ; v13=v1*v13*v3; v31=v13'
            ; v23=x1'[w23]x1; v23=v23(1:k,1:k); v23=v2*v23*v3 ; v32=v23' $
MATRIX      ; If(j >= 4) | v04=x1'[w04]x1 ; v04=v04(1:k,1:k)
            ; v04=v0*v04*v4; v40=v04'
            ; v14=x1'[w14]x1; v14=v14(1:k,1:k); v14=v1*v14*v4 ; v41=v14'
            ; v24=x1'[w24]x1; v24=v24(1:k,1:k); v24=v2*v24*v4 ; v42=v24'
            ; v34=x1'[w34]x1; v34=v34(1:k,1:k); v34=v3*v34*v4 ; v43=v34' $
MATRIX      ; If[j >= 1] | v0=v0(1:k,1:k) ; v1=v1(1:k,1:k) ; v=[v0,v01/v10,v1] $
MATRIX      ; If[j >= 2] | v2=v2(1:k,1:k)
            ; v=[v0,v01,v02 / v10,v1,v12 / v20,v21,v2] $
MATRIX      ; If[j >= 3] | v3 =v3(1:k,1:k)
            ; v=[v0,v01,v02,v03/v10,v1,v12,v13/v20,v21,v2,v23/v30,v31,v32,v3] $
MATRIX      ; If[j >= 4] | v4 =v4(1:k,1:k)
            ; v=[v0,v01,v02,v03,v04/v10,v1,v12,v13,v14/
                 v20,v21,v2,v23,v24/v30,v31,v32,v3,v34/v40,v41,v42,v43,v4] $
```

Compute the Wald statistic.

MATRIX ; db = d*bt ; dvd = d * v * d' $
MATRIX ; List ; Brant = db' * <dvd> * db $
CALC ; List ; df = Col(dvd) ; Ctb(.95, (Col(v))) ; l = Chi(Brant,df) $

R16.5 Reading Matrices

You can import matrices from other data sources for example from a spreadsheet program such as *Excel* or as text in an ordinary data file.

R16.5.1 Importing a Matrix as a Data File

Since a matrix of values looks the same as a data set, you can read one directly from a file or from your screen. The command looks the same as that for reading data into variables in your data area but the specifications are for a matrix instead. Thus,

IMPORT ; Rows = the number of rows
; Cols = the number of columns
; Matrix = the name of the matrix you are reading $

You can create a new matrix in this way, or you can replace an existing one. If you replace a matrix, the new dimensions can be different from the old one. An example appears below. The **IMPORT** command imports the Longley data into the matrix *longley*.

```
IMPORT ; Rows = 16 ; Cols = 7 ; Matrix = Longley $
1947  83.0 234289 1590 60323 8256 38407
1948  88.5 259426 1456 61122 7960 39241
1949  88.2 258054 1616 60171 8017 37922
1950  89.5 284599 1650 61187 7497 39196
1951  96.2 328975 3099 63221 7048 41460
1952  98.1 346999 3594 63639 6792 42216
1953  99.0 365385 3547 64989 6555 43587
1954 100.0 363112 3350 63761 6495 42271
1955 101.2 397469 3048 66019 6718 43761
1956 104.6 419180 2857 67857 6572 45131
1957 108.4 442769 2798 68169 6222 45278
1958 110.8 444546 2637 66513 5844 43530
1959 112.6 482704 2552 68655 5836 45214
1960 114.2 502601 2514 69564 5723 45850
1961 115.7 518173 2572 69331 5463 45397
1962 116.9 554894 2827 70551 5190 46652
```

The two limits on the command are

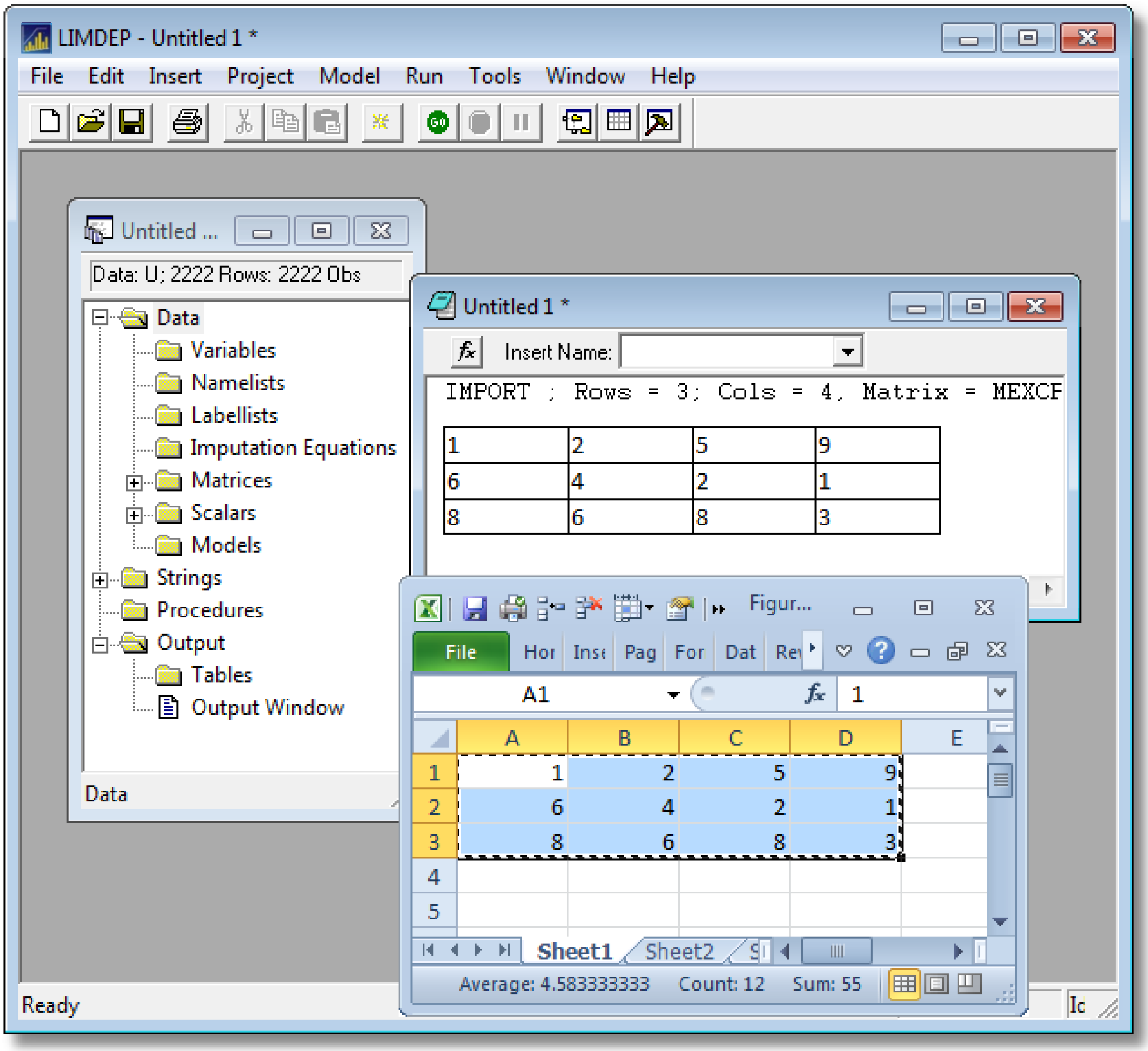
- *Rows*×*Cols* must be less than or equal to 50,000.
- You may only read one matrix in an **IMPORT** command. To read more than one matrix, just use a separate **IMPORT** command for each one.

R16.5.2 Importing a Matrix as a Block of Cells from *Excel*

A block of cells in a spreadsheet program such as *Excel* may be transported directly into a named matrix. Use the text editor as described above and the following steps:

1. Create a template **IMPORT** command. Put the **IMPORT** command in the text editor first.
2. Use edit/copy in *Excel* to copy the rectangular block of cells.
3. Use edit/paste in *LIMDEP* to put the cells in the editor below the **IMPORT** command.
4. Submit the command as usual by highlighting it and clicking GO**.**

Figure R16.9 shows an example. Note that the grid markers are transferred into the text editor with the matrix. These will be ignored by the command processor when the data are imported.



Data: U; 2222 Rows: 2222 Obs

```
IMPORT ; Rows = 3; Cols = 4, Matrix = MEXCF
```

1	2	5	9
6	4	2	1
8	6	8	3

	A	B	C	D	E
1	1	2	5	9	
2	6	4	2	1	
3	8	6	8	3	
4					
5					

Average: 4.583333333 Count: 12 Sum: 55

Figure R16.9 Exporting a Matrix to *LIMDEP* from *Excel*

R16.6 Matrix Expressions

Most of the operations you do with matrices, particularly if you are constructing estimators, will involve expressions, products, sums, and functions such as inverses. This section will show how to arrange such mathematical expressions of matrices. We have used these procedures at many points in our earlier discussion.

As noted above, every numerical entity in *LIMDEP* is a matrix and may appear in a matrix expression. There are very few functions that require data matrices. These will be noted below. The algebraic operators are

*****	for matrix multiplication,
+	for addition,
-	for subtraction,
’ (apostrophe)	for transposition and also for transposition then multiplication,
/	for a type of division (see below),
^	for raising a matrix to a power (several forms, see below).

Thus, **c*d** equals **C**×**D** and **c*Ginv(c)** (or **c*<c>**) equals **C** times its inverse, or **I**, and **c′*Ginv(c)*c** equals **C′**. As will be evident shortly, the apostrophe operator, (**’**) is a crucial part of this package.

When scalars appear in matrix computations, they are treated as scalars for purposes of computation, not as matrices. Thus, **A***s***B′**, where *s* is a scalar, is the same as *s***AB′**. The 1×1 matrix in the middle does not interfere with conformability; it produces scalar multiplication. 1×1 matrices which are the result of matrix computations, such as quadratic forms, also become scalars for purposes of matrix multiplication. Thus, in **A′ * r’b*r * A** will not require conformability of **A’** and **r’** (number columns of **A′** equal number of rows of **r′**) if the quadratic form **r’Br** is collected in one term; also, **A′*r′*B*r*A** does require conformability, but the same expression could be written **a′ * r’[B]r * a** to achieve greater efficiency. If **r** happens to be a variable, this may be essential. The implications of these different forms will be presented in detail below.

All syntaxes are available for any entity, so long as conformability is maintained where appropriate. **A** and **B** are any matrix; **w** is any vector, row or column, including, if desired, a variable; and, **C** is any matrix. (Once again, a matrix is any numeric entity – there is no need to distinguish, e.g., variables from previously computed matrices.)

Each result in the following table produces a result that, for later purposes, may be treated as a single matrix.

a’b	=	transpose of *a* times *b*
a’[*w*]*b*	=	*a*’diag(*w*) *b* (Do not create diagonal matrices!)
a’< *w*> *b*	=	*a*’[diag(*w*)]$^{-1}$ *b*
a’[*c*] *b*	=	*a*’ *c b* = bilinear form
a’< *c*> *b*	=	*a*’ *c*$^{-1}$ *bc* is any matrix.
< *a*>	=	*a*$^{-1}$
[*a*]	=	*G*-2 inverse of *a*.
< *a*’ *b*>	=	(*a*’ *b*)$^{-1}$
< *a*’[*w*] *b*>	=	(*a*’[*w*] *b*)$^{-1}$
< *a*’< *w*> *b*>	=	(*a*’< *w*> *b*)$^{-1.}$

Table R16.1 Matrix Expressions

In a matrix expression, the symbol ‘1’ can be used where needed to stand for a column of ones. Thus,

1**’***a* = a row of ones times matrix *a*
*a***’**1 = transpose of matrix *a* times a column of ones.

Note that in each of these cases, the apostrophe is an operator that connotes multiplication after transposition.

NOTE: You should never need to compute *a*′ ***** *b*. Always use *a*′*b*. Thus, in the earlier example, *c*′< *c*> *c* is better than *c*′*****Ginv(*c*)***** *c* or *c*′< *c*>***** *c*.

These functions will be particularly important for using matrix algebra with large amounts of data. Section R16.7 gives further details.

In any matrix function list, you may use the transpose operator for transposition. For example, two ways to obtain the sum of a matrix and its transpose are

sum = *a* + *a***’** and *sum* = Msum(*a***,** *a*’).

The transpose of a matrix may appear in an expression simply by writing it with a following apostrophe. For example,

a’*c*’*ca* could be computed with *a*’ ***** *c*’ ***** *c* ***** *a*

though *a*’ ***** *c*’*c* ***** *a* would be necessary if *c* were a data matrix.

You may string together as many matrices in a product as desired. As in the example, the terms may involve other matrices or functions of other matrices. For example, the following commands will compute White’s heteroscedasticity corrected covariance matrix for the OLS coefficient vector.

```
NAMELIST    ; x = list of Rhs variables $
REGRESS     ; Lhs = y ; Rhs = x ; Res = e $
CREATE      ; esq = e^2 $
MATRIX      ; white = <x’x> * x’[esq]x * <x’x>
```

The **CREATE** command that computes the squared residuals is actually unnecessary. The last two lines could be combined in

```
MATRIX      ; white = <x’x> * Bhhh(x,e) * <x’x> $
```

LIMDEP also provides a function to compute the center matrix for the Newey-West estimator;

Nwst(*x*,*e*,*l*) computes the Newey-West middle matrix for *l* lags. *l* = 0 => White.
e is the vector of residuals, *x* is a namelist defining the set of variables.

You may also multiply simple matrices that you enter directly. For example

$$\begin{bmatrix}1 & 3\\3 & 5\end{bmatrix}\begin{bmatrix}2 & 4\\5 & 5\end{bmatrix}$$ = **[1 / 3,5] * [2,4 / 5,5]**.

The multiplication operator sorts out scalars or 1×1 matrices in a product. Consider, for example, $\mathbf{V} = \mathbf{A}(\mathbf{r}'\mathbf{A}\mathbf{r})^{-1}\mathbf{A}'$.If **r** is a column vector, this is a matrix (**AA′**) divided by a quadratic form. To compute this, you could use

```
MATRIX        ; v = a * <r'[a]r> * a' $
```

The scanner will sort out scalars and multiply them appropriately into the product of matrices. But, in all cases, matrices which are not 1×1 must be conformable for the multiplication. Thus, if **r** were a matrix instead of a vector, it might not be possible to compute **V**.

The '+' operator is used to add matrices. Thus, to add the two matrices above instead of multiply them, we could use

$$\begin{bmatrix}1 & 3\\ 3 & 5\end{bmatrix} + \begin{bmatrix}2 & 4\\ 5 & 5\end{bmatrix} \;=\; \textbf{[1 / 3,5] + [2,4 / 5,5]}.$$

The matrix subtraction operator is '**-**.' Thus, *a* **-** *c* gives **A - C** (of course).

You may also combine the **+**, **-**, and ***** operators in a command. For example, the restricted least squares estimator in a classical regression model, when the linear restrictions are **Rb = q**, is

$$\mathbf{b}_r = \mathbf{b}_u - (\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}(\mathbf{R}\mathbf{b}_u - \mathbf{q}).$$

This could be computed with

```
NAMELIST    ; x = list of variables $
MATRIX      ; bu = <x'x> * x'y
            ; r = ... ; rt = r'
            ; q = ...
            ; d = r * bu - q
            ; xxi = <x'x>
            ; br = bu - xxi * r' * <rt'[xxi]r > * d $
```

(Why did we transpose *r* into *rt* then use *rt***'**, which is just *r*, in the last expression? Because the apostrophe operator is needed to produce the correct matrix multiplication inside the <> operation. There are other ways to do this, but the one above is very convenient.) Notice in the preceding that if there is only one constraint, *r* will be a row vector, and the quadratic form will be a scalar, not a matrix.

Any of the product arrangements shown in Table R16.1 may appear in any function or expression as if it were already an existing matrix. For example, Root(<*x*'[*w*]*x* >)computes the characteristic roots of $[\Sigma_i w_i \mathbf{x}_i \mathbf{x}]^{-1}$. But, longer matrix expressions may not be grouped in parentheses, nor may they appear as arguments in other matrix functions. Expressions which must be used in later sums and differences or functions must be computed first. There will usually be other ways to obtain the desired result compactly. For examples,

	d = Sinv(*a* + *c*) is invalid but can be computed with	*d* = Nvsm(*a*,*c*),
	d = (*a* + *c*) ***** *q* is invalid but can be computed with	*d* = Msum(*a*,*c*) ***** *q*,
and	*d* = Ginv(*a* ***** *q* ***** r) is invalid, but can be computed with	*d* = Iprd(*a*,*q*,*r*).

The functions Msum, Mdif, Nvsm, and Iprd may facilitate grouping matrices if necessary.

R16.6.1 Scalar Multiplication of a Result – Using CALCULATE

You can multiply the result you are obtaining by a scalar by using

name = *value* * *expression*

For example, one way to create a 2×2 identity matrix would be *i*2 **=** Iden(2). Then,

*teni*2 = 10 * Iden(2)

would create a multiple of the identity matrix.

As with **CALC** and **CREATE**, the names *n* and *pi* are always interpreted as the current sample and the number 3.14159..., respectively. Thus, you might compute $n(\mathbf{X'X})^{-1}$ by using *n* ***** <*x'x*>. A scalar multiple may be a number, a scalar value you have computed earlier, or an element of an existing matrix, such as *r*(3,4), For example, after computing a regression, since *ssqrd* is kept automatically,

vb = *ssqrd* ***** <x'x>

would reproduce the calculation of *varb* done automatically by the regression. As a shortcut, *LIMDEP* also allows simple scalar division of a matrix. The syntax can be of these forms

name = 1 / *value* * *expression* *or* *name* = <*value*> * *expression*.

For example, the following three expressions are equivalent:

a **=** **.**1 ***** Iden(5) = 1/10 * Iden(5) = <10> * Iden(5).

The method of obtaining more complicated scalar multiples is given below. Note from the list in Table R16.1 that you can use <*s*> ***** *A* for 1/*s* ***** A since <*s*> implies inversion, then the 1×1 is detected.

You can issue a **CALCULATE** command from a **MATRIX** command if you enclose the command in curled brackets. It might look like this:

```
MATRIX      ; {1 - 2 * Phi(1.96)} $
```

which does not save anything since dropping the brackets and changing **MATRIX** to **CALC** does the same thing. However,

```
MATRIX      ; {p = 1 - 2 * Phi(1.96) ; s2 = sumsqdev/n } * <x'x> $
```

is convenient – it computes *p*, then *s*2 which is then used to compute the variance of the maximum likelihood estimator of the coefficient vector in a regression model.

This gives the same result as giving this command with **CALC**, instead. You can put any valid **CALCULATE** command in the brackets. It can be as complex as necessary, with subcommands separated by semicolons. The advantage of this is that you can use the result as a scalar multiple for a matrix result by preceding your **MATRIX** command with the **CALCULATE** command. For example, the following computes $n(2\pi)^{-1/2}$ times an $(\mathbf{X}'\mathbf{X})^{-1}$ matrix:

{*n*/(Sqr(2**pi*))} * Xpxi(*one*,*age*,*exper*).

It is also possible to keep the scalar result at the same time as the matrix result. For example, the preceding could be modified to

qxxi = {*v* = *n*/(Sqr(2**pi*))} * Xpxi(*one*,*age*,*exper*).

The scalar *v* is calculated inside the curled brackets. When it is obtained, the matrix *qxxi* is computed as *v* times the inverse of the **X′X** matrix. This command computes a scalar and a matrix result at the same time.

If your **CALC** command has more than one subcommand and you use it for scalar multiplication, the value sent back as the scalar in the curled brackets is the last value calculated. For example, the following would compute the '*teni*2' obtained above:

{1 + 1 ; *abc* = 3 ; *r* = 10} * Iden(2)

R16.6.2 Adding the Same Scalar to Every Element of a Matrix

A common operation in econometrics is adding the same value to each element of a matrix. Consider, for example, adding one to every element of matrix **A**. The general operation is done with

MATRIX ; a = [value] + a $

More generally, whenever a 1×1 matrix, or a scalar, is added to another matrix, the operation is taken to mean that the scalar should be added to (or subtracted from) every element of the larger matrix. Thus, all of the following are valid commands:

a = [*pi*] + a
a = *b*'<*varb*>*b* + *a*

(which adds the same quadratic form to every element of *a*),

a = *a* + [1] *q* + [2]

and so on.

R16.6.3 Raising a Matrix to a Power

There are several ways to raise a matrix to a power. Let **A** denote an $r \times c$ matrix, **P** denote a $q \times s$ matrix, and *d* denote a scalar: The matrices **A** and **P** may be the names of existing matrices, or any construction of a matrix that produces a matrix result, including functions. Thus, one possibility is

result = Sqrt(<*x*'[*w*]*x*>) ^ 2 (which equals <*x*'[*w*]*x*>).

Matrix to Scalar Power

A ^ *d* raises the matrix to the *d* power. **A** must be the name of a square matrix. The following cases are allowed:

1. *d* = 0 returns identity matrix for all **A**.

2. *d* = positive integer. Repeated multiplication.

3. *d* = negative integer. Same as $(\mathbf{A}^{-1})^d$. **A** must be nonsingular.
   Thus, *a*^-2 is the same as Ginv(*a*)^2 and *a*^-1 is the same as Ginv(*a*), the ordinary inverse.

4. *d* = positive real number. Returns the spectral decomposition,
   $\mathbf{Q} = \mathbf{C}\mathbf{\Lambda}^d\mathbf{C}'$ where **C** is the matrix of columns of characteristic vectors and **Λ** is the diagonal matrix of characteristic roots. **A** must be symmetric.

5. *d* = negative real number. Same as case 4 except that all roots must be positive.
   I.e., **A** must be positive definite. This replaces the Sqrt function (now, *result* = *a*^.5) and the Isqr function (now, *result* = *a*^-.5).

Matrix to Matrix Power

A ^ **P**. **A** and **P** must have the same dimensions, but need not be square. Then, element by element, this returns **A**(*i,j*) ^ **P** (*i,j*) = [**A**(*i,j*) ^ **P**(*i,j*)].

Scalar to Matrix Power

d ^ **A**. **A** is any matrix. Each element of the result is *d* raised to the power of the corresponding element of **A**. Thus, the dimensions of the result are those of **A**.

Matrix to Scalar Power, Element by Element

A ! *d* raises each element of *a* to the *d* power. Note that this differs completely from **A**^ *d*. For example, the following is valid, if a bit farfetched:

```
CALC      ; q = Log(Phi(1.2*rsqrd)) $
MATRIX    ; v = <x'x> ! q $
```

R16.6.4 Entering, Moving, and Rearranging Matrices

To define a matrix, use

MATRIX ; name = [... row 1 / ... row 2 ... / ...] $

Elements in a row are separated by commas while rows are separated by slashes. For example,

MATRIX ; a = [1,2,3,4 / 4,3,2,1 / 0,0,0,0] $ creates $a = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 4 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$.

To facilitate entry of matrices you can use these two arrangements:

k | value	= a $K\times 1$ column vector with all elements equal to value
*k*_value	= a $1\times K$ row vector with all elements equal to value

Thus, in the last row above, 0,0,0,0 could be replaced with 4_0.

Symmetric matrices may be entered in lower triangular form. For example,

MATRIX ; a = [1 / 2 , 3 / 4 , 5 , 6] $ creates $a = \begin{bmatrix} 1 & 2 & 4 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix}$.

Matrix elements given in a list such as above may be scalars, or even other matrices and vectors. For example, to compute the column vector, $[\boldsymbol{\gamma}' , \theta] = [(1/\sigma)\boldsymbol{\beta}',(1/\sigma)]'$ after fitting a tobit model, you could use

```
TOBIT     ; ... $
CALC      ; theta = 1/s $
MATRIX    ; gamma = theta * b ; gt = [gamma / theta] $
```

Note that the slash used here indicates stacking, not division, and that *gt* is a column vector.

Partitioned Matrices

A partitioned matrix may be defined with submatrices. For example, suppose *c*1 is a 5×2 matrix and *c*2 is 5×4. The matrix *c* = [*c*1,*c*2] is a 5×6 matrix which can be defined with

MATRIX ; c = [c1 , c2]

The two matrices must have the same number of rows. Matrices may also be stacked if they have the same number of columns. For example:

To obtain $\mathbf{F} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{bmatrix}$ use **f = [c1 / c2]**.

To obtain $\mathbf{M} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix}$ use **m = [m11,m12 / m21,m22]**.

Symmetric matrices may be specified in lower triangular form. For example, suppose **M** were symmetric, so that **M21** = **M12′**. **M** could be constructed using

m = [m11 / m21 , m22].

The application of the Brant test in Section R16.4.5 gives an extensive example that uses partitioned matrices.

Block Diagonal Matrix

Form a block diagonal matrix from scalars and/or square matrices with

a = Blkd(*c*1, *c*2, …, *ck*).

Matrices *c*1,…,*ck* may be any mix of scalars and square matrices.

Matrices with Identical Elements

If a matrix or vector has all elements identical, use *a* = Init(*r*,*c*,*s*). This initializes an *r*×*c* matrix with every element equal to scalar, *s*. This is a way to define a matrix for later use by an estimation program. Example 3 in Section R16.1 shows an application. This method can also be used to initialize a row (*r* = 1) or column (*c* = 1) vector. Alternatively, you could use *a* = [*c*_*s*] for a row vector or *a* = [*r*|*s*] for a column vector. The function Ones(*r*) returns an *r*×1 column vector of ones. For a vector of ones, you can use

Ones(*k*) = *k*×1 column of ones.

Identity Matrices

To define an *r*×*r* identity matrix, use *a* = Iden(*number of rows*). It may be useful to use a scalar for the number of rows. For example, suppose that *x* is the name of a namelist of *K* variables which will vary from application to application and you will require *a* to be a *K*×*K* identity matrix, where *K* is the number of variables in *x*. You can use the following:

```
CALC      ; k  = Col(x) $
MATRIX    ; ik = Iden(k) $
```

The notation I[r] produces the same matrix as Iden(r). To extract a specific row or column from an identity matrix, use the functions

Iden(*k*,*j*)	= the *j*th column of *K*×*K* identity matrix, also Evec(*k*,*j*)
Iden(*k*,-*j*)	= the *j*th row of the same matrix.

Band Matrices

Certain applications in time series analysis, for example the calculation of the approximations to the distribution of the Durbin-Watson statistic, require a band matrix. This has zeros everywhere except above and below the principal diagonal, which are ones. To create such a matrix, use

a = Iden (- *number of rows*)

For example,

$$\text{Iden}(-3) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Also, **i[-r]** may be used for the band matrix with ones on the sub and super diagonal of an $r \times r$ matrix. This is the same as Iden(-*r*).

Random Matrices

The **MATRIX** command *a* **=** Rndm(*list*)can be used to draw matrices of random numbers from the normal distribution. The following specifications may be used:

Rndm(*m*)	=	$m \times 1$ random vector from standard normal,
Rndm(*r*,*m*)	=	$r \times m$ random matrix from standard normal.

All elements are independent draws. You may also specify the mean vector and covariance matrix for a draw of a random vector from the normal distribution:

Rndm(*mu*)	=	$r \times 1$ random vector from normal distribution with mean *mu* and covariance matrix **I**. The matrix *mu* may be a row or column vector, and *r* is the number of elements in *mu*.
Rndm(*sigma*)	=	$r \times 1$ random vector from multivariate normal distribution with mean vector **0** and covariance matrix *sigma*. The number of rows in *sigma* is *r*. You must provide a positive definite *sigma* matrix.
Rndm(*mu*,*sigma*)	=	$r \times 1$ random vector from multivariate normal distribution with mean vector *mu* and covariance matrix *sigma*. The matrix *mu* must be the name of a row or column vector with *r* elements, and *sigma* must be the name of a square matrix with *r* rows.

Halton draws are discussed in Section R24.7. You can use **CREATE** to obtain columns of Halton draws. **MATRIX** will create the set of Halton draws for the first *K* prime numbers in each row of a matrix using the function Hltn.

Hltn(*n*,*K*)	**=**	*n* creates a matrix whose *n* rows are Halton draws. There are *K* columns using the first *K* primes for bases of the Halton sequences

Nodes and Weights for Hermite and Laguerre Quadrature

To obtain a display of the sets of nodes and weights used for Gaussian quadrature, you can use the function

Quad(*n*,*H*) or Quad(*n*,*L*)	to produce a listing of quadrature points, weights and nodes for *n* points, *H* = Hermite, *L* = Laguerre

The 20 point Hermite quadrature values are listed below.

```
Result  |            1            2
--------+--------------------------
       1|      -5.38748   .222939E-12
       2|      -4.60368   .439934E-09
       3|      -3.94476   .108607E-06
       4|      -3.34785   .780256E-05
       5|      -2.78881   .228339E-03
       6|      -2.25497     .00324377
       7|      -1.73854      .0248105
       8|      -1.23408       .109017
       9|      -.737474       .286676
      10|      -.245341       .462244
      11|       .245341       .462244
      12|       .737474       .286676
      13|       1.23408       .109017
      14|       1.73854      .0248105
      15|       2.25497     .00324377
      16|       2.78881   .228339E-03
      17|       3.34785   .780256E-05
      18|       3.94476   .108607E-06
      19|       4.60368   .439934E-09
      20|       5.38748   .222939E-12
```

The Hermite quadrature weights and nodes can be accessed from the matrix and used to compute integrals of the form

$$\int_{-\infty}^{+\infty} \exp(-x^2)\, f(x)dx \approx \sum_{h=1}^{H} weight_h\, f(node_h)$$

The weights appear in the first column and the nodes are given in the second. The counterparts for Gauss-Laguerre quadrature are used when the limits of integration are (0,+∞) and the weighting function is Exp(-x). Built in functions that automate these procedures may also be used in **MAXIMIZE/MINIMIZE**. See Chapter E66.

Multivariate Normal Probabilities

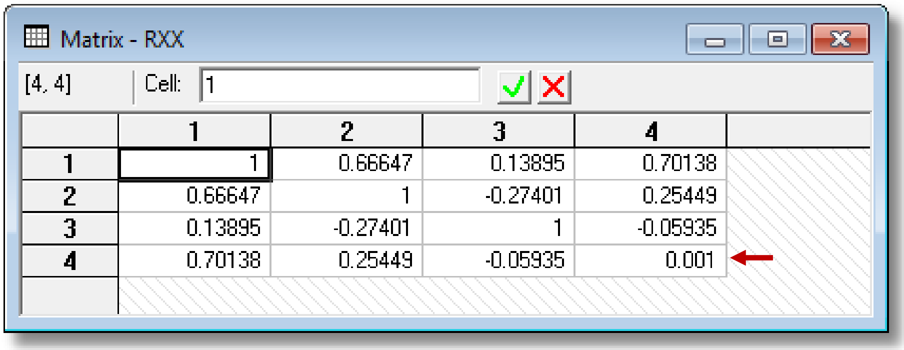
The matrix function Mvnp produces a column vector of probabilities from the multivariate normal CDF. The syntax is Mvnp(*x*,*w*)where *w* is a *J*×*J* covariance matrix of the random vector. The mean vector is assumed to be zero. *x* is either a matrix or a namelist of variables, either with *J* columns. Each row is taken as the vector from the multivariate normal distribution. Thus, if *x* has *K* rows, the result of Mvnp(*x*,*w*) is a *K* element column vector with *k*th element equal to the multivariate normal CDF evaluated at *w* and the corresponding row of *x*. The function Mvnd(*x*,*w*) returns the multivariate normal density, rather than the CDF.

Editing a Matrix

Replace an element in a matrix with

matrixname (*i*) = *value* or *matrixname* (*i*, *j*) = *value*.

The display window for a matrix also allows you to edit the matrix. In the example in Section R16.2.2, we noted that the (4,4) matrix in the matrix in the display window in Figure R16.10 should be



	1	2	3	4
1	1	0.66647	0.13895	0.70138
2	0.66647	1	-0.27401	0.25449
3	0.13895	-0.27401	1	-0.05935
4	0.70138	0.25449	-0.05935	0.001

Figure R16.10 Matrix Editing Window

1.0. With the display on the screen, you can change a matrix just by changing the value in the cell in the display. The change will be recorded in the matrix stored in memory. (The display is the actual matrix, not a copy of it.)

Equating One Matrix to Another

Use *a* = *c* to equate *a* to *c*. To equate *a* to the transpose of *c*, use *a* = *c***'**. You would typically use this operation to keep estimation results. After each model command, the estimated parameter vector is placed in the *read only* matrix, *b*. Thus, to avoid losing your coefficient vector, you must equate something to *b*.

Extracting Part of a Matrix

To extract a submatrix from matrix *c*, use *a* = Part (*c*, *r*1, r2, *c*1, *c*2); *a* is the submatrix of *c* consisting of rows *r*1-*r*2 and columns *c*1-*c*2. If *c* is *a*, this will discard some of the rows and columns of **A**. If **C** is a vector, you may omit the superfluous pair of subscripts. Thus,

a = Part (*c*, 1, 5)

extracts elements 1 through 5 of vector *c*. If *c* is a column vector, *a* will be also. If *c* is a row vector, *a* will be a row vector.

The Part function can be abbreviated as follows: If *a* is a row or column vector with *K* elements, *a* (*first* : *last*) denotes the subvector of '*length*' elements of *a* beginning with the '*first*.' E.g., *a*(4:6) is elements 4, 5, and 6 of *a*. *The result is always a column vector, even if* a *is a row vector to begin with.* This may be used wherever a set of values is desired, for example, in lists of starting values, in initializing or extracting from matrices with the **MATRIX** command, lists of limits for LDV models, etc. For a two dimensional matrix, you may use *a*(*r*1:*r*2, *c*1:*c*2) instead of Part (*c*, *r*1, *r*2, *c*1, *c*2).

To extract parts of rows or columns of a matrix, use

name = *vector*(-*j*)	= vector without element *j*
name = *matrix* (-*j*,-*m*)	= matrix without row *j* and column *m*
name = *matrix* (*j*,-*m*)	= row *j* without element m
name = *matrix* (-*j*,*m*)	= column m without element *j*

Injecting Vectors into Matrices

The statement

name (*, *j*) = *vector*

replaces column *j* of the matrix with the elements of the vector. The '*****' indicates 'replace all rows in column *j*.' You may also replace a row with

name (*j*, *) = *vector*.

Finally, to replace the diagonal elements of the matrix with the elements of the vector, use

name (*, *) = *vector*.

The vector on the right may be a row or column; it is just treated as a string of numbers. Also, the matrices on the left and right need not be conformable. For example, in the first case, if the vector has more elements than there are rows in the matrix, then some elements of *vector* will be left over and discarded. If, on the other hand, *name* has more rows than there are elements in *vector*, then the column will only be partially replaced. The row replacement works the same way. For the third construction, the matrix need not be square, and, once again, may have dimensions different from *vector*. The replacement is done for the principal diagonal, (*i*,*i*) elements, until either there are no more rows or columns in the matrix or until the entire vector has been moved. If *name* is a vector, you may use *J* = 1 to overlay part of one vector with another. The row or column indicator may be given in a scalar as well as a value. For example,

```
MATRIX      ; sampleb = Init(10,2,0) $
PROC $
CREATE      ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
REGRESS     ; Lhs = y ; Rhs = one, x $
MATRIX      ; sampleb(i,*) = b $
ENDPROC $
EXECUTE     ; Silent ; i = 1,10 $
```

This places 10 vectors of least squares slopes into the rows of matrix *sampleb*.

Inserting One Matrix into Another

The construction

LhsMatrix (*i*,*j*) = *RhsMatrix*

puts the Rhs matrix into the Lhs matrix with upper left corner of the Rhs matrix at location (*i*,*j*) of the Lhs matrix. This may also be used to put vectors into vectors, or vectors into matrices (but not matrices into vectors). Row and column dimensions are strictly enforced. The operation is ignored if either dimension of the Rhs matrix would go past the corresponding boundary of the Lhs matrix.

Redimensioning a Matrix

The function

$$a = \mathrm{Vctr}(d)$$

makes $r \times c$ matrix d into $1 \times rc$ row vector a by running the rows in order into a vector. Specific forms of this operation are

$a = \mathrm{Runc}(d)$	equivalent to Vctr(d')
$a = \mathrm{Stkc}(d)$	equivalent to transpose of Vctr(d)
$a = \mathrm{Stkr}(d)$	equivalent to transpose of Vctr(d').

The reverse operation would be

$$a = \mathrm{Mvec}(d,r,c)$$

This takes any matrix which has r times c elements in any arrangement, and produces an r by c matrix.

Column Vector from Symmetric Matrix

The function

$$a = \mathrm{Vech}(c)$$

of $k \times k$ symmetric matrix c returns $k(k+1)/2 \times 1$ column vector a from the lower triangle of c. For example,

$$a = Vech\begin{pmatrix} 1 & 4 \\ 4 & 9 \end{pmatrix} = \begin{pmatrix} 1 \\ 4 \\ 9 \end{pmatrix}.$$

Diagonal Elements of a Matrix in a Vector

The function

$$a = \mathrm{Vecd}(c)$$

creates column vector a from the diagonal elements of the matrix c.

Diagonal Matrix Created from a Vector

The command

$$a = \mathrm{Diag}(c)$$

creates a square matrix a with diagonal elements equal to those of row or column vector c. If c is a square matrix, the diagonal elements of a will be the same as those of c while the off diagonal elements of a will be zero.

R16.7 Using MATRIX Commands with Data

LIMDEP's matrix package is designed to allow you to manipulate large amounts of data efficiently and conveniently. Applications involving up to three million observations on 150 variables are possible. With **MATRIX**, manipulation of a data matrix with 1,000,000 rows and 50 columns, which would normally take 400 megabytes of memory just to store, is not only feasible, but no more complicated than it would be if the data set had only 100 rows instead! It is important for you to be aware of how this is done in order to use this program successfully.

The essential ingredient is the form in which matrix results generally appear in econometrics. It is quite rare for an estimator or a procedure to be based upon 'data matrices,' per se. Rather, they almost always use functions of those matrices, typically moments, i.e., sums of squares and cross products. For example, an OLS estimator, $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$, can be viewed as a function of $\mathbf{X}$ and $\mathbf{y}$. But, it is much more useful to view it as a function of $\mathbf{X}'\mathbf{X}$ and $\mathbf{X}'\mathbf{y}$. The reason is that, regardless of the number of observations in the data set, these matrices are $K\times K$ and $K\times 1$, and K is usually small. *LIMDEP* uses this result to allow you to manipulate your data sets with matrix algebra results, regardless of the number of observations. To underscore the point, consider that currently, most other econometrics packages provide a means of using matrix algebra. But, to continue our example, in order to do a computation such as that for **b** directly, some of them must physically *move* the data that comprise **X** into an entity that will be the matrix, **X**. Thus **X** must be created, *even though the data used to make* **X** *are already in place, as part of the data set currently being analyzed.* It is this step which imposes the capacity constraints on some econometrics programs. Avoiding it allows *LIMDEP* to manipulate data matrices of any length. The utility of this approach will be clear shortly.

It is important to keep in mind the distinction between two kinds of matrices that you will be manipulating. We define them as follows:

- **Data matrices:** A data matrix is a set of rows defined by observations and columns defined by variables. The elements of the data matrix reside in your data area which is discussed in Sections R2.8 and R3.4.
- **Computed matrices:** A computed matrix is the result of an operation that is based on data matrices or other computed matrices. The elements of a computed matrix will reside in your matrix work area, which is defined below.

The distinction is purely artificial, since, as will soon be evident, every numeric entity in *LIMDEP* is a matrix. The important element is that *the size of a data matrix is* $n\times K$ *where* n *is the current sample size and* K *is a dimension that you will define. The size of a computed matrix is* $K\times L$ *where* K *and* L *are numbers of variables, or some other small values that you will define with your commands.*

R16.7.1 Data Matrices

To use your data to compute matrices, you will usually define 'data matrices.' This amounts to nothing more than labeling certain areas of the data array; you do not actually have to move data around (whatever that might mean) to create a data matrix. For *LIMDEP*'s purposes, a data matrix is any set of variables which you list. You can overlap the columns of data matrices in any way you choose; data matrices may share columns. An example appears below. One useful shortcut which can be used to 'create' a data matrix is simply to associate certain variables and observations with the matrix name by using **NAMELIST**. The variables are defined with the **NAMELIST** command.

The rows or observations are defined by the current sample, with the **SAMPLE**, **REJECT/INCLUDE**, **DRAW**, and **PERIOD** commands. If you change the current sample, the rows of all existing data matrices change with it. If you change the variables in a namelist, you redefine all matrices that are based on that namelist.

For example, suppose the data array consists of the following:

```
YEAR      CONS    INVST     GDP    PRICES
1995      1003      425    1821     124.5
1996      1047      511    2072     139.2
1997      1111      621    2341     154.7
1998      1234      711    2782     177.6
```

Two data matrices, *demand* and *all data* would be defined by the command

NAMELIST ; demand = cons,invst,gnp
; alldata = year,cons,invst,gdp,prices $

Notice that these data matrices share three columns. In addition, any of the 31 possible subsets of variables can be a data matrix, and all could exist simultaneously.

The number of rows each data matrix has depends on the current sample. For example, to have the matrices consist of the last three rows of the data, it is necessary only to define

SAMPLE ; 2-4 $

You can vary the sample at any time to redefine the data matrices. For example, suppose it is desired to base some computations on *demand* using all four years, and then compute other matrices using *alldata* only for the last three years. The sequence might appear as follows:

SAMPLE ; All $
MATRIX commands using *demand*
SAMPLE ; 2-4 $
MATRIX commands using *alldata*

The reason for the distinction between data and computed matrices is this: Consider the computation of a matrix of weighted sums of squares and cross products

$$\mathbf{F} = (1/n)\mathbf{X}'\mathbf{W}\mathbf{X}$$

where $\mathbf{X}$ is $n \times K$ with n being the sample size, and $\mathbf{W}$ is an $n \times n$ diagonal matrix of weights. Suppose n were 10,000 and K were 20. In principle, just setting up $\mathbf{X}$ and $\mathbf{W}$ for this computation would require at least $8(10000 \times 20 + 10000 \times 10000)$, or over 800 million bytes of memory, before computation even begins! But, computations of this size are routine for *LIMDEP*, because

- $\mathbf{F} = (1/n)\Sigma_i w_i \mathbf{x}_i \mathbf{x}_i'$ where $\mathbf{x}_i$ is a row of $\mathbf{X}$, which is always only 20×20, and
- The data needed for the sum already exist in your data area.

That is, by treating this sort of computation as a summing operation, not as a formal matrix product, we can achieve tremendous efficiencies. The important feature to exploit is that regardless of n, the result will *always* be $K \times K$.

R16.7.2 Computations Involving Data Matrices

You can manipulate any sized data matrix with **MATRIX**. There are two simple rules to remember when using large samples:

- Ensure that in any expression, **MATRIX ; name = result $**, the target matrix (*name*) is not of the order of a data matrix. That is, neither rows nor columns is *n*. This will be simple to achieve, since the sorts of computations that you normally do will ensure this automatically.
- Ensure that when data matrices appear in an expression, they are either in the form of a moment matrix, i.e., in a summing operation, or they appear in a function that does summing.

Suppose that *x* and *y* are data matrices defined as above with 500,000 rows and 25 columns each (i.e., they are very large). Any operation that uses *x* or *y* directly will quickly run into space problems. For example,

```
MATRIX      ; z = x’ * y $
```

(the matrix product equal to the transpose of *x* times *y*) is problematic, since copies of both *x*’ and *y* must be created. But, the apostrophe is a special operator, and

```
MATRIX      ; z = x’y $
```

can be computed because in this form *LIMDEP* knows that the operation is a sum of cross products, and will be only 25×25. An alternative way to obtain the *z* above is

```
MATRIX      ; z = Xdot(x,y) $
```

The second rule above, then, amounts to this: When data matrices appear in matrix expressions, they should always be in some variant of *x’y*, i.e., as an explicit sum, or in one of the special moment functions listed in Section R16.9, such as Xdot. The apostrophe (**’**) is a special operator in this setting. Although it can be used in multiplying any matrices, it is the device which allows you to manipulate huge data matrices, as illustrated by the examples given at the beginning of this chapter. All of them work equally well with small samples or huge ones. The commands are all independent of the number of observations.

Finally, consider the weighted sum above, **F** = (1/*n*)**X′WX** where there are 10,000 observations and 20 variables. Once again, the result is going to be 20×20. *LIMDEP* provides many different ways to do this sort of computation. For this case, the best way to handle it is as follows:

```
NAMELIST    ; x  =  the list of 20 variables $
SAMPLE      ; ... set up the 10,000 observations $
CREATE      ; w =  the weighting variable  $
MATRIX      ; f  = 1/n * x’[w]x  $
```

This would work with 10 or 10,000,000 observations. The matrix *f* is always 20×20.

The following program transcript illustrates the manipulation of a moderately large data set.

```
--> RESET
--> LOAD     ; file = "...healthcare.lpj" $
    Project file contained   27326 observations.
--> SAMPLE   ; All $
--> NAMELIST ; x = one,age,educ,married $
--> CREATE   ; y = married ; Weight = Exp(.23*female) $
--> MATRIX   ; bw = <x'[weight]x> * x'[weight]income $
--> CREATE   ; ew = y - x'bw $
--> MATRIX   ; vw = {ew'ew/(n-col(x))} * <x'[weight]x> $
--> MATRIX   ; Stat(bw,vw,x) $
```

```
-----------------------------------------------------------------------------
Number of observations in current sample =   27326
Number of parameters computed here       =       4
Number of degrees of freedom             =   27322
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  Matrix|  Coefficient       Error       z     |z|>Z*        Interval
--------+--------------------------------------------------------------------
Constant|     .04834***      .00676     7.15   .0000      .03509    .06158
     AGE| .27115D-04      .8686D-04      .31   .7549 -.14312D-03  .19735D-03
    EDUC|     .02120***      .00042    50.28   .0000      .02038    .02203
 MARRIED|     .08277***      .00227    36.47   .0000      .07832    .08721
--------+--------------------------------------------------------------------
Note: nnnnn.D-xx or D+xx => multiply by 10 to -xx or +xx.
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The data set contains 27,326 observations. The matrix commands compute a weighted least squares linear regression of y on x. In principle, the weighting matrix in square brackets is 27,327 × 27,326. However, the data matrix, **X**, and two variables, *y* and *weight*, are already defined in the data area, so the **MATRIX** commands do not reproduce the data. None of the matrices produced by the **MATRIX** commands are larger than 4×4.

R16.8 Functions for Manipulating Matrices

The preceding described how to operate the matrix algebra package. The examples in Section R16.1 also showed some of the more common uses of **MATRIX**. This and the following sections will now detail the specifics of *LIMDEP*'s matrix language. In addition to the basic algebraic operations of addition, subtraction, and multiplication, *LIMDEP* provides nearly 100 different functions of matrices, most of which can, themselves, be manipulated algebraically. The matrix functions can be combined with the algebraic operators to create matrix expressions. Some of these, such as Nvsm(.) are used to combine algebraic results, while others, such as Root(.) are specialized functions that produce complex transformations of matrices. Any of the constructions in Table R16.1 can be used as a standalone matrix. For example, to obtain the determinant of $(\mathbf{X'WX})^{-1}$, where **W** is a diagonal weighting matrix, you can use Dtrm(<*x*'[*w*]*x*>). Likewise, several such constructions can appear in functions with more than one input matrix. This should allow you to reduce some extremely complex computations to very short expressions.

R16.8.1 Functions of One Matrix

Square Roots

a = Sqrt(c)	- **AA** = **A**^{2} = **C**, same as **A ^ .5**,
a = Isqr(c)	- **AA** = **A**^{2} = **C**$^{-1}$, same as **A ^ -.5**,
a = Orth(c)	- **A** = **C** × **Isqr**(**C′C**) (orthonormalizes **C**, **A'A** = **I**),
a = Proj(c)	- **A** = **C**(**C′C**)$^{-1}$**C′**.

Characteristic Roots and Vectors

a = Cvec(c) - characteristic vectors.

If **C** is a K×K matrix, **A** has K columns. The kth column is the characteristic vector which corresponds to the kth largest characteristic root, ordered large to small. **C** must be a symmetric matrix. If not, only the lower triangle will be used.

a = Root(c) - characteristic roots of a symmetric matrix.

For symmetric matrix **C**, **A** will be a column vector containing the characteristic roots ordered in descending order. For nonsymmetric matrices, use

a = Cxrt(c) - possibly complex characteristic roots of asymmetric matrix.

The characteristic roots of a nonsymmetric matrix may include complex pairs. The result of this function is a K×2 matrix. The first column contains the real part. The corresponding element of the second column will be the imaginary part, or zero if the root is real. The roots are ordered in descending order by their moduli.

You can use Cxrt to obtain the dominant root for a dynamic system. Then, the modulus can be obtained with **CALC**. Let **C** be the relevant submatrix of the structural coefficient matrix in the autoregressive form. Then,

```
MATRIX    ; rt = Cxrt(C) $
CALC      ; check = rt(1,1)^2 + rt(1,2)^2 $
```

Cxrt(c) gives the same results as Root(c) if **C** is a symmetric matrix. But, if **C** is nonsymmetric, Root(c) gives the wrong answer because it assumes that **C***is* symmetric, and uses only the lower triangle.

It is always possible to obtain the roots of a symmetric matrix. But, certain nonsymmetric matrices may not be decomposable. If this occurs, an error message results.

The Root function can also be used to find the possibly complex roots of a dynamic equation. If **C** is a vector of elements [c_1 ,c_2 ,...,c_Q] instead of a symmetric matrix, then A = Root(c) reports in a K×2 matrix the reciprocals of the characteristic roots of the matrix

$$\mathbf{C} = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_Q \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

These are the roots of the characteristic equation, $1 - c_1 z - c_2 z^2 - ... - c_Q z^Q = 0$, of the dynamic equation

$$y_t = c_1 y_{t-1} + c_2 y_{t-2} + ... + c_Q y_{t-Q} + \textit{other terms}.$$

The dominant root of the system is the largest reciprocal reported. If its modulus is larger than one, the equation is unstable.

Cholesky Decomposition

A positive definite matrix, **C**, can be factored into the outer product of a lower triangular matrix, **L**. I.e., **C** = **LL′**. The function

a = Chol(c)	- lower triangular matrix of Cholesky decomposition

returns the matrix **L**. An alternative representation is $\mathbf{A} = \mathbf{L}^*\mathbf{D}\mathbf{L}^{*\prime}$, where **D** is the diagonal matrix of Cholesky values. The elements of **D** are the squares of the diagonal elements in **L**, so,

$$\mathbf{L} = \mathbf{L}^* \times \mathbf{D}^{1/2}$$

To extract **D** and $\mathbf{L}^*$ from **L** you could use

MATRIX ; l = Chol(a) ; d = Diag(l) * Diag(l) ; ls = l * Isqr(d) $

Singular Value Decomposition

An $m \times K$ matrix **C** for which $m > K$ may be reduced to its singular value decomposition

$$\mathbf{C} = \mathbf{U} \times \mathbf{D} \times \mathbf{V}'$$

where **U** is $m \times m$ with $\mathbf{U}'\mathbf{U} = \mathbf{I}$, **V** is $K \times K$ with $\mathbf{V}'\mathbf{V} = \mathbf{I}$, and **D** is a diagonal matrix containing the singular values of **C**. Among its other virtues, the singular value decomposition is useful for accurate and fast inversion, as $(\mathbf{C}'\mathbf{C})^{-1} = \mathbf{V}*(\mathbf{D}'\mathbf{D})^{-1}*\mathbf{V}'$. Use the function

a = Svdx(c)	- singular value decomposition, partitioned as $\begin{bmatrix} \mathbf{V} \\ \mathbf{D} \end{bmatrix}$

to obtain this decomposition. This function returns a $2K \times K$ matrix. The first K rows contain **V** while the second K rows contain the diagonal matrix, **D**. The leading matrix, **U**, is not returned. (The matrix **U** is $m \times m$. If you are decomposing a data matrix with, say, 10,000 rows, **U** would be 10,000×10,000. **U** is generally not needed in subsequent computations; the useful information about the moments of the data matrix will be contained in **V** and **D**.) This function also returns a scalar named *svd_rank* which contains column rank of **C**. The rank is simply the number of nonzero singular values. Some related functions are

Svdd(c)	- column vector of singular values
2nrm(c)	- 2-norm of c = largest singular value
2cnm(c)	- singular value rank = largest singular value/smallest (if positive)

Element By Element Transformations

a = Loge(*c*)	- element by element natural log,
a = Expn(*c*)	- element by element exponent,
a = Diri(*c*)	- direct inverse = element by element reciprocal,
a = Esqr(*c*)	- element by element square root,
a = Sign(*c*)	**-** gives matrix of signs of the elements of **C**, -1,0,1 for -,0,+.

These functions return zero for elements for which they cannot be computed.

The matrix power functions listed earlier can also be used to transform matrices element by element. In particular,

a = *c* ! *q*	- element of **C** raised to the *q* power.
a **=** *c* ^ *d*	- if **C** and **D** have the same dimensions, $\mathbf{A}_{ij} = \mathbf{C}_{ij} \wedge \mathbf{D}_{ij}$.

Inverse Matrices

LIMDEP computes three types of inverse matrices. For a nonsingular and not necessarily symmetric matrix, **C**, you can use

a = Ginv(*c*) - $\mathbf{A} = \mathbf{C}^{-1}$.

The inverse of a matrix may also be written in the form*a* **=** *c*^-1. The method used is pivoting and row reduction. If the matrix to be inverted is symmetric and positive definite, a faster procedure which uses the Cholesky decomposition is

a = Sinv(*c*) - $\mathbf{A} = \mathbf{C}^{-1}$.

Use this for regression problems and inversion of moment matrices. If the matrix to be inverted is symmetric but short ranked, two types of generalized inverses are available. Let **C** be the matrix to be inverted. A G-2 inverse is the **A** such that

$$\mathbf{A} = \mathbf{ACA} \text{ and } \mathbf{C} = \mathbf{CAC}.$$

For this matrix, you can use

a = G2nv(*c*) - G2 inverse of square matrix

The Moore-Penrose inverse is a G-2 inverse which also satisfies the requirement that **ACA** and **CAC** be symmetric. To compute it, use

a = Mpnv(*c*) - Moore-Penrose inverse

The Moore-Penrose inverse is computed as

$$\text{Mpnv}(c) = \Sigma_i(1/\lambda_i)\mathbf{c}_i\mathbf{c}_i'$$

where λ_i is a nonzero characteristic root of **C** and $\mathbf{c}_i$ is the associated characteristic vector.

Two additional functions are provided for inverting sums and products. The matrix function Nvsm gives the inverse of the sum of the matrices in the list in parentheses. Thus,

; Nvsm(x1, x2, a, q) = $(\mathbf{X}_1+\mathbf{X}_2+\mathbf{A}+\mathbf{Q})^{-1}$.

and **; Nvsm(<x'x>, <y'[z]y>) =** $[(\mathbf{X}'\mathbf{X})^{-1} + (\mathbf{Y}'\mathbf{W}\mathbf{Y})^{-1}]^{-1}$.

Matrices in the sum may also carry minus signs. Thus,

; Nvsm(x1, x2, -a) = $(\mathbf{X}_1 + \mathbf{X}_2 - \mathbf{A})^{-1}$.

An inverse of a product of matrices is computed with

; Iprd(c1, c2, ...) = $(\mathbf{C}_1 \times \mathbf{C}_2 ...)^{-1}$.

The computation of the restricted least squares estimator in Example 2 in Section R11.1 gives an example in which

; Iprd(r, <x'x>, r') = $[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}$.

Scalar Functions

The following always result in a 1×1 matrix:

a = Dtrm(*c*)	- determinant of square matrix,
a = Logd(*c*)	- log-determinant of positive definite matrix,
a = Trce(*c*)	- trace of square matrix,
a = Norm(*c*)	- Euclidean norm of vector **C**,
a = Norm(*c*)	- norm of matrix **C** = square root of trace of **C′C,**
a = 2nrm(*c*)	- 2 norm of matrix **C** = largest singular value of **C**,
a = Rank(*c*)	- rank of any matrix.

The rank is computed as the number of nonzero characteristic roots of **C'C**. To find the rank of a data matrix **X** (i.e. several columns of data in a namelist, **X**), you could use

c = Rank(*x*).

However, this may not be reliable if the variables are of different scales and there are many variables. You should use, instead,

MATRIX ; C = Diag(X'X) ; C = Isqr(C) * X'X * Isqr(C) ; Rank(C) $

R16.8.2 Functions of Two or More Matrices

a = Kron(*c*, *d*)	- $\mathbf{A} = \mathbf{C} \otimes \mathbf{D}$	(Kronecker product)
a = Dirp(*c*, *d*, ...)	- $\mathbf{A} = [\mathbf{C}_{ij}\mathbf{D}_{ij} ...]$	(direct product).

The direct product, or Hadamard product, **A**, of two matrices **C** and **D** is $\mathbf{A}_{ij} = \mathbf{C}_{ij}\mathbf{D}_{ij}$. **C** and **D** may be the same matrix. You may multiply any number of matrices with this command. All must have the same dimensions. For example, **c = Dirp(c,c,c,c)** replaces each element of **C** with its own fourth power.

a = Qrow(*c*, *d*) - column vector of quadratic forms.

The Qrow function returns a column vector. **C** and **D** are conformable matrices for the product **A** = **CDC’**. Qrow returns the diagonal elements of *a* in a column vector. Each element is the corresponding quadratic form of the row in **C** and the matrix **D**.

a = Msum(*c*, *d*, ...)	- matrix sum (same as **C** + **D** **...**),
a = Mdif(*c*, *d*)	- matrix difference, **C**- **D**.

Msum and Mdif may be used to group matrices in a sum or difference for multiplication. I.e., *c* * (*d*+*e*) * *c* is not valid, but *c* ***** Msum(*d*,*e*) ***** *c* is, and would be equivalent.

R16.9 Sums of Observations

There is no obstacle to computing a matrix **X′X**, even if **X** has 1,000,000 rows, so long as the number of columns in **X** is not more than 225. The essential ingredient is that **X′X** is not treated as the product of a $K\times n$ and an $n\times K$ matrix, it is accumulated as a sum of $K\times K$ matrices. By this device, the number of rows, *n*, is immaterial (except, perhaps, for its relevance to how long the computation will take). Matrix operations that involve **C′AC** or $\mathbf{C}'\mathbf{A}^{-1}\mathbf{C}$ are, as in all cases, limited to 50,000 cells. But, suppose that **C** is 5,000×2 and **A** is a diagonal matrix. Then, the result is only 2×2, but apparently it cannot be computed because **A** requires 25,000,000 cells. But, in fact, only 5,000 cells of **A** are needed, those on the principal diagonal. *LIMDEP* allows you to do this computation by providing a vector (in this case, 5,000×1) instead of a matrix, for a quadratic form. Thus, in **c’[a]c**, if **a** is a column or row vector, *LIMDEP* will expand (at least in principle) the diagonal matrix and compute the quadratic form **C′AC** as if **A** were Diag(*a*). This result will be crucial when **C** is a data matrix, **X**, which may have tens or hundreds of thousands of rows. The second aspect of the computation of matrices that involve your data is that once the data are in place in the data area, in fact, there is no need to create **A** or Diag(*a*) at all. The data are just used in place; you need only use variables and namelists by name.

Invariably, when you manipulate data matrices directly in matrix algebra expressions, you will be computing sums of squares and/or cross products, perhaps weighted, but in any event, of order $K\times K$. The simple approach that will allow you to do so is to ensure that when *xnames* and *vnames* (namelists and variables) appear in matrix expressions, they appear in one of the following constructions, where *x* and *y* are namelists of variables and *w* is a variable: Some data summation functions are listed in Table 16.2.

x’x	=	the usual moment matrix.
x’y	=	cross moments.
x’[*w*] *x*	=	**X’diag(w)X**, weighted sums. **w** is a variable. Or, **X’[w]Y**.
x’<*w*> *x*	=	$\mathbf{X'[diag(w)]^{-1}X}$, weighted by reciprocals of weights.
< *x’ x*>	=	$\mathbf{(X'X)}^{-1}$, inverse of moment matrix.
< *x’y*>	=	$\mathbf{(X'Y)}^{-1}$, inverse of cross moments, if it exists .
< *x’*[*w*] *x*>	=	$\mathbf{(X'[w]X)}^{-1}$, inverse of weighted moments. Or <**X’[w]Y**>.
< *x’*<*w*> *x*>	=	$\mathbf{(X'<w>X)}^{-1}$, inverse, weighted by reciprocals of weights.

Table R16.2 Sums of Observations in Matrix Functions

Again, the use of the apostrophe operator here is important in that it sets up the summing operation that allows you to use large matrices. That is, while logically **x’ * y** is the same as **x’y** for *LIMDEP*’s purposes, they are very different operations. The left hand side requires that copies of *x* and *y* be made in memory, while the right hand side requires only that the sum of cross products be accumulated in memory.

TIP: All sample moments are computed for the currently defined sample. If the current sample includes variables with missing data, you should make sure the **SKIP** switch is turned on. Missing values in a matrix sum are treated as valid data, and can distort your results. If you precede the **MATRIX** command(s) with **SKIP**, then in summing operations, observations with missing values will be ignored.

Matrix functions that compute sums from the data may operate on subsets of the data as follows:

```
MATRIX      ; For [variable = value] ; the matrix command $
```

For example, suppose *x* is a namelist and *female* is a dummy variable. The matrix command

```
MATRIX      ; For [female = 1] ; xx = x’x $
```

computes the sums of squares and products for *x* using only observations for which *female* equals one. This does not change the current sample; it relates only to the matrix instruction itself.

NOTE: The setting of the subsample does not apply to any function that changes the data, such as Pcom (principal components) or to any panel data operations such as Xtxp.

To illustrate, the following extends Example 1 from Section R16.1 to constrained weighted least squares. The matrix *x* is the namelist of right hand side variables, *y* is the name of the left hand side variable, and *w* is a variable which contains the weights (variances). (Note, 1/w_i appears in the summations.)

```
MATRIX      ; xwxi = <x’<w>x>
            ; bu = xwxi * x’<w>y
            ; d = r * bu - q
            ; h = Iprd(r,xwxi,r’)
            ; br = bu - xwxi * r’ * h * d $
CREATE      ; u = y - x’br $
CALC        ; df = n - Col(x) + Row(r) $
MATRIX      ; s2 = 1/df * u’<w>u
            ; vr = s2 * xwxi - s2 * xwxi * r’ * h * r * xwxi $
```

The operations described here for manipulating data matrices are logically no different from other matrix operations already described. That is, in your expressions, there is no real need to distinguish data manipulations from operations involving computed matrices. The purpose of this section is to highlight some special cases and useful shortcuts.

Sums, Means, and Weighted Sums of Observations and Subsamples

To sum the rows of a data matrix, use

; name = x’1 or **x’one**

The symbol, **1** is allowable in this context to stand for a column of ones of length *n*. This returns a $K\times 1$ column vector whose *k*th element is the sum of the *n* observations for the *k*th variable in *x*. To obtain a row vector instead, use

; name = 1’x or **one’x**

Do note, in most applications, this distinction between row and column vectors will be significant. You can obtain a sample mean vector with

; name = 1/n * x’1

A matrix function, Mean, is also provided for obtaining sample means, so

; Mean(x) = 1/n * x’1

Note that the Mean function always returns a column vector of means, so if you want a row, you must transpose the column after using the function. (**1/n*1’x** may be more convenient.) The Mean function provides one advantage over the direct approach. You can use Mean with a list of variables without defining a namelist. Thus, to obtain the means of *z,x,w,*log*(k), f*21, you could use

; name = Mean(z, x, w, Log(k), f21).

To obtain a weighted mean, you can use

; name = <1’w> * x’w

where *w* is the weighting variable. Note that this premultiplies by the reciprocal of the sum of the weights. If the weights sum to the sample size, then you can use **1/n** or **<n>** instead of **<1’w>**. A related usage of this is the mean of a subsample. To obtain a mean for a subsample of observations, you will need a binary variable that equals one for the observations you want to select and zero otherwise. Call this variable *d*. Then,

; name = <1’d> * x’d

will compute the desired mean. The Mean function also allows weights, so you can use Mean(*x*,*w*) or Mean(*x*,*d*). In order to use this construction, the parameters of the Mean function must be a namelist followed by a variable.

Covariances, Correlations, and Standard Deviations

You can use the formulas for variances directly to obtain covariance matrices, but the functions,

	Xvcm(*x*)	- covariance matrix for **X**
	Xcor(*x*)	- correlation matrix for **X**
and	Ktau(list of variables)	- Kendall's tau matrix form

will probably be simpler. Kendall's τ is a nonparametric measure of the correlation between two variables, defined as

$$\tau_{xy} = \frac{\Sigma_{i=1}^{n}\Sigma_{j=1}^{i-1}\operatorname{sgn}(x_i, x_j)\operatorname{sgn}(y_i, y_j)}{n(n-1)/2}$$ where Sgn(x_i,x_j) = +1 if $x_i > x_j$ and -1 if $x_i \le x_j$.

Each of these may be weighted in the same fashion as the Mean function. I.e., Xvcm(*x*,*w*) and Xcor(*x*,*w*) compute the covariances and correlations with a weighting variable. As before, *w* can be replaced with a subsampling indicator (binary variable), *d*.

Another construction allows you to obtain covariances and correlations for two sets of variables:

Xvcm(*x*,*y*)	- cross covariance for **X** and **Y**
Xcor(*x*,*y*)	- cross correlation for **X** and **Y**.

Note that Xvcm(x,y)$_{ij}$ = Cov(x_i,y_j) and likewise for the correlations. That is, the first namelist defines the row variables in the matrix and the second defines the columns.

A vector of standard deviations is requested with

Sdev(*x*)	- column vector of standard deviations of variables in namelist.

You may also provide weights in Sdev(*x*,*w*). For extracting correlations or standard deviations from a covariance matrix, use

Mcor(*v*)	- converts a covariance matrix **V** to a correlation matrix.
Msdv(*v*)	- creates a diagonal matrix with standard deviations on the diagonal.
Vsdv(*v*)	- extracts a column vector of standard deviations from covariance matrix **V** by arraying in the vector the square roots of the diagonal elements of matrix **V**.
Mdcr(*x*)	- computes the matrix of correlations from a data matrix defined by a namelist *x*.
Mdvc(*x*)	- computes the matrix of covariances from a data matrix defined by a namelist *x*.

The Mdcr and Mdvc functions may also compute cross correlations and covariances for two lists of variables by using (*x*,*y*) where *y* is a second namelist.

Sums of Squares and Cross Products

Matrices of the form **X′X** are obtained directly, as **x'x** = $\Sigma_i \mathbf{x}_i \mathbf{x}_i'$ and **x'[w]x** = $\Sigma_i w_i \mathbf{x}_i \mathbf{x}_i'$. All of the other constructions involving inversion, **<x'x>**, weighting by reciprocals, and so on, apply here. These were listed in Table R16.2. As in the previous cases, the weighting variable may be a binary indicator for a subsample.

For using a list of variables that is not collected in a namelist, you can use the Xdot function. Thus,

; Xdot(x) = x'x

and so on for the weighted variants. But, suppose the data set consists of variables *x*1,*x*2,*x*3. Then,

; Xdot(x*) = Xdot(x1,x2,x3) = x'x

might also be used (if there were no other variables). In addition, **xx = Xdot(*)** would be the same as **xx = Xdot(one,x1,x2,x3)**. Some other possibilities would be

```
      ; xxlogs = Xdot(Log(x1),Log(x2),Log(x3)) $
and   ; auto   = Xdot(x1,x1[-1],x1[-2],x1[-3]) $
```

Note that, as earlier, in this format, you cannot use a weighting scheme, as there is no way to distinguish a weighting variable from simply the last variable in the list. Some additional functions that may be used are

Xpxi(...)	- $(\mathbf{X'X})^{-1}$
Xcpm(...)	- $1/n\ \mathbf{X'X}$
Xcpi(...)	- $n(\mathbf{X'X})^{-1}$

These functions would be useful for computing the moment matrices for a list of variables, but the direct formula, e.g., **<x'x>**, will be preferable for a namelist.

Some additional formats may also be specified. To obtain cross moment matrices, any of the preceding may be specified with a pair of namelists instead and/or a weighting variable. Thus,

A = Xdot(x,y,w) = **x'[w]y** = $\Sigma_i w_i \mathbf{x}_i \mathbf{y}_i'$,

where column vector $\mathbf{x}_i$ is the transpose of row *i* of **X** and, as usual, the first namelist defines the row variable.

Sums of Squares and Cross Products of Deviations

Use the shorthands

$$\mathbf{x}'[1]\mathbf{x} \;=\; \sum_{i=1}^{n} (\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})'$$

with matrix **X** defined by the namelist *x*, and similarly for **x**'[1]**y** for namelist or variable y.

Panel Data Sums of Squares

In general, the computation is

$$A = \sum_{i=1}^{n} \sum_{t=1}^{T_i} \left(\mathbf{x}_{it} - \theta_i \overline{\mathbf{x}}_i\right)\left(\mathbf{z}_{it} - \theta_i \overline{\mathbf{z}}_i\right)$$

When $\theta_i = 1$ and $\mathbf{x} = \mathbf{z}$, this is the within group sum of squares. The two data vectors may be different. The value θ may be a single number that you specify. For computing the GLS estimator for a random effects linear model, you would use $\theta = 1 - \sigma_u/[\sigma_v^2 + T_i\sigma_u^2]^{1/2}$ and you would supply σ_v and σ_u. The function requires *x* and *z* to be defined either by a namelist or a variable. Then,

a = Xtxp(*x*)	=	within group sum of squares for *x*
a = Xtxp(*x*,*z*)	=	within group sum of cross products,
a = Xtxp(*x*,*z* | θ)	=	within group sum of squares or cross products partial difference from mean
a = Xtxp(*x*,*z* | σ_v,σ_u)	=	GLS form of matrix.

If the result of the computation is a square matrix, the function

a^{-1} = Xtxi(…) = inverse matrix.

For example, if you have σ_v and σ_u in hand, the GLS coefficient vector for the linear random effects model would be

$\mathbf{b}_{gls}$ = Xtxi(X,X | σ_v,σ_u) * Xtxp(X,y | σ_v,σ_u).

NOTE: These two functions must be preceded by a **SETPANEL** command that sets the dimensions of the panel data set for the program.

Outer Product Matrices

Expressions for asymptotic covariance matrices are often of the form $\mathbf{A} = \Sigma_i z_i^2 \mathbf{x}_i \mathbf{x}_i'$. This is the usual format for the BHHH estimator. To obtain **A**, you can use

```
CREATE      ; z2  =  z^2 $
MATRIX      ; a  =  x'[z2]x  $
```

A function that makes the **CREATE** command unnecessary is

a = Bhhh(*x*, *z*).

For partitioned matrices such as

$$\mathbf{A} = \begin{bmatrix} \sum_{i=1}^{n} z_i^2 \mathbf{x}_i \mathbf{x}_i' & \sum_{i=1}^{n} z_i w_i \mathbf{x}_i \mathbf{y}_i' \\ \sum_{i=1}^{n} w_i z_i \mathbf{y}_i \mathbf{x}_i' & \sum_{i=1}^{n} w_i^2 \mathbf{y}_i \mathbf{y}_i' \end{bmatrix},$$

you can use the function

a = Bhhh(x, y, z, w).

where x and y are namelists and z and w are the weighting variables, to obtain the result without constructing the parts separately. The inverse of the resulting matrix is

a^{-1} = Bhhh(x,y,z,w).

This form of moment matrix arises frequently in index function models. For example, for the probit model, the vector of first derivatives of the log likelihood for the *i*th observation is

$\mathbf{g}_i = z_i\mathbf{x}_i$ where $z_i = q_i\phi(\boldsymbol{\beta}'\mathbf{x}_i)/\ \Phi(q_i\boldsymbol{\beta}'\mathbf{x}_i)$ and $q_i = 2y_i - 1$.

Thus, the BHHH, or OPG estimator of the asymptotic covariance matrix for the MLE of $\boldsymbol{\beta}$ is of the form above. Indeed, if the estimate of $\boldsymbol{\beta}$ is in hand, the OPG estimator could be easily found as follows:

```
CREATE      ; z = Lmd((b’x),(1-y)) $
MATRIX      ; opg = Bhhh(x,z)  $
```

(Note the necessary sign switch in the Lmd function to obtain the derivatives.)

Least Squares Computations

To obtain a matrix or vector of least squares regression coefficients, you can use

; bols = <x’x> * x’y.

y can be a single variable, in which case this produces a column vector of regression coefficients, or *y* can be a namelist to produce a matrix of coefficients whose *j*th column is the coefficients in the regression of the *j*th column of *y*on all of the columns of *x*. Weights for weighted least squares are provided in the usual fashion;

; bwls = <x’[w]x> * x’[w]y

and subsamples may be drawn as well with binary variables.

The function Xlsq(*x*,*y*) or Xlsq(*x*,*y*,*w*) can be used for the same computations. If a list of variable names is given in

; bls = Xlsq(list)

the *last* variable name is taken to be the ‘*y*’ in a least squares regression. With this construction, it is not possible to use weights.

The sum of squares and cross products of the residuals in a regression of *y* on **X** is

$$\mathbf{e'e} = \mathbf{y'y} - \mathbf{y'X}(\mathbf{X'X})^{-1}\mathbf{X'y}.$$

You can obtain this with

; ee = y'y - y'x * <x'x> * x'y.

But it will be simpler to use the function

$$\text{Rcpm}(x,y) = \mathbf{y'y} - \mathbf{y'X}(\mathbf{X'X})^{-1}\mathbf{X'y}.$$

To save a step in some analyses, you can invert this matrix with

$$\text{Rcpi}(x,y) = (\mathbf{y'y} - \mathbf{y'X}(\mathbf{X'X})^{-1}\mathbf{X'y})^{-1}.$$

Certain specification tests in econometrics require a matrix of the form Rcpm or Rcpi using weights of the form given in the Bhhh function above. To use this form of weighting in these functions, use Rcpm(*x*,*y*,*z*,*w*) or Rcpi(*x*,*y*,*z*,*w*). Note, again, with this form, the weights are squares or cross products of z_i and w_i.

The condition number of a data matrix, **X**, is the square root of the ratio of the largest to smallest characteristic root of the scaled moment matrix, $[\text{diag}(\mathbf{X'X})]^{-1/2}\mathbf{X'X}\,[\text{diag}(\mathbf{X'X})]^{-1/2}$. Use the function

Cnum(*x*)	- the condition number for data matrix *x* defined by a namelist.

The 2-norm of a data matrix is the largest singular value, which can be computed using

2nrm(x)	- largest singular value of matrix *x*.

Least Absolute Deviations

The matrix function

Ladb(*x*,*y*)	- median regression (least absolute deviations) estimator,

where *x* is a namelist and *y* is a variable, produces the least absolute deviations coefficient vector. You can also obtain the coefficient vector for a different quantile with the **QREG** model command.

Heteroscedasticity and Autocorrelation Robust Covariance Matrices

The following commands will compute White's heteroscedasticity corrected covariance matrix for the OLS coefficient vector.

```
NAMELIST   ; x = list of Rhs variables $
REGRESS    ; Lhs = y ; Rhs = x ; Res = e $
CREATE     ; esq = e^2 $
MATRIX     ; white = <x'x> * x'[esq]x * <x'x> $
```

The **CREATE** command that computes the squared residuals is actually unnecessary. The last two lines could be combined in

```
MATRIX      ; white = <x'x> * Bhhh(x,e) * <x'x> $
```

There is a single function that combines both of these.

Nwst(*x*,*e*,*l*)	computes the Newey-West middle matrix for *l* lags; *e* is the vector of least squares residuals and *x* is a namelist defining the set of variables.

Setting *l* equal to zero gives the White estimator. Thus, combining results, we have, for the White estimator,

```
MATRIX      ; white = <x'x> * Nwst(x, e, 0) * <x'x> $
```

while the Newey-West estimator for, say, 10 lags is

```
MATRIX      ; neweywst  =  <x'x> * Nwst(x, e, 10) * <x'x>  $
```

R16.10 Matrix Commands that Transform the Data

The preceding discussion has detailed matrix operations that are functions of other matrices and of the data in the data area. None of the operations described so far actually change your data in any way. Some operations on the data themselves are much more convenient with matrix algebra than, for example, with **CREATE** commands. The following functions modify the raw data – do note, these changes are permanent.

R16.10.1 Linear Transformations of Variables

The following functions replace *x* with a transformation of *x*, where *x* is a namelist that defines a data matrix. In each case, the original data are lost. If you wish to retain them, it is necessary to create copies of the variables in *x*. To use a column of ones in any of these, it is necessary to create one. Thus, *x* must be defined by a **NAMELIST** command, and may not contain the variable *one*.

x = Xmlt(*s*) - scalar multiplication of the data matrix.

Every item in the data matrix is multiplied by the scalar, *s*, either a number, a scalar, or a matrix element.

x = Xmlt(*c*) - linear combination of the variables

In this function, *c* is a square matrix with the same number of columns as *x*. Each variable in *x* is replaced with the linear combination of all of the columns of *x* defined by the corresponding column of *c*.

x = Xmlt(*v*) - rowwise multiplication by variable *v*.

This multiplies all variables in a row by a variable in the row. Each variable in *x* is multiplied by the corresponding observation of the variable *v*. The variable *v* should not be in *x*. Note that this is the same as the **CREATE** function Scl(*x*,*v*) except that Xmlt(*v*) does not create a new namelist.

x = Indx(*y*,*c*) - linear function of a data matrix.

Here, *y* is another data matrix; *c* is a matrix with the same number of columns as *x* and number of rows equal to the number of columns in *y*. Each column of *x* is replaced by the linear combination of the variables in *y* defined by the corresponding column of *c*. The namelists *x* and *y* must already be defined as namelists. If *y* and x have variables in common, this will produce unpredictable (and no doubt, undesirable) results.

In general, no parameter list is required for the following functions. Each operation is controlled by the currently defined sample. As before, *x* must be defined by a **NAMELIST** command. In the function,

; x = Xstd

the columns of *x* are standardized by subtracting the column mean, then dividing by the standard deviation.

; x = Xdev

The columns of *x* are centered by subtracting the column mean, but are not rescaled.

; x = Xorn

The columns of *x* are orthonormalized. The data matrix, *x*, is replaced by $x^* = \mathbf{X}(\mathbf{X}'\mathbf{X})^{1/2}$. Finally,

; x = Pcom

The columns of *x* are replaced by the principal components of **X**, ordered by their contribution to the total variation in *x* (the trace of **X′X**). The linear transformations are the characteristic vectors of **X′X**, normalized to unit length.

Keep in mind when using these functions that the variables only appear once in the data array. When you change them, you change all data matrices which contain these variables. A **MATRIX** command which operates on the data affects (or uses) only those observations included in the current sample. Thus, for example, if you were to use a subset of your observations to compute some principal components, the unused rows would be unchanged after the computation.

R16.10.2 Moving a Matrix into the Data Area

It should never be necessary to move data from your data area to your matrix workspace – data matrices are already manipulable as if they were computed matrices. But, the reverse move might be useful. For example, after you compute a fixed effects regression using panel data, the estimated fixed effects are stored in a matrix named *alphafe*. You might be interested in regressing *alphafe* on a set of variables, or computing a set of descriptive statistics for some other purposes. In order to do so, you will have to be able to access *alphafe* as if it were a variable. A means of doing so is to use a simple **CREATE** command,

```
CREATE      ; x = alpha $
```

(This operation must be the only one done with this **CREATE** command. I.e., **CREATE ; w = 1 ; x = m $** is invalid.) That is, you simply equate the namelist to the matrix.

The operation proceeds as follows: Suppose *x* has *K* columns (i.e., the namelist defining *x* has *K* variables). However, *K* may be one and *x* may be the name of a variable. The current sample has *n* observations, so *n* is the number of rows in *x*. Let *a* be any $P\times Q$ matrix in the matrix work area. The command moves as many rows and columns of *a* into *x* as possible, starting at the top of the sample and at the left in the list of variables. Note that the rows are as defined by the current sample, whatever that may be. If *x* is a variable name and *a* is a matrix, elements of *x* are replaced by the first column of *a*. If *a* is a row vector, it has only one row. A matrix is *not* transposed by a copy. Thus, you cannot move a row vector into a column of your data array; you must create the column vector first by transposing the matrix to be copied.

R16.11 MATRIX Commands for Panel Data

There are numerous transformations and matrix manipulations that are specifically written for panel data. Chapter R5 described some of these. This section will provide further detail on some matrix functions that can be used to manipulate panels. For most of this, you will require either a ‘stratification’ variable that identifies groups in a panel with their own code or group number or identification, or a ‘group count’ variable such as generally used in **; Pds** in the various model commands.

There are two sets of procedures provided for manipulating panel data. A number of matrix functions are provided for computing group aggregates in panel data sets. For example, with a panel data set with, say, 1,000 individuals each observed 11 times for a sample of 11,000 observations, you might want to manipulate the 1000×1 vector of group means. You might also want to manipulate the individual observations in a panel. For example, for some purposes (e.g., random effects models), you might want a variable, $z_{it} = \bar{z}_i$ that, for individual *i*, equals the mean of the T_i observations for that individual. That is, the group mean is repeated for the T_i observations. The **CREATE** commands below are used for that purpose.

R16.11.1 MATRIX Functions for Panel Data

The index variable described in the previous section gives the group identifiers in a panel data set. In addition to the estimation programs, there are several matrix functions provided specifically for panel data. Each of these requires you to provide a group indicator. The functions are:

Grps(*variable*,*index*)	= a five column matrix of group sizes, means, standard deviations, minima, and maxima, with one row for each group. There may be empty rows.
Gsiz(*index*)	= a column vector of group sizes,
Gxbr(*list*,*index*)	= an $N\times K$ matrix of group means; N = number of groups, K = number of variables,
Gsdv(*list*,*index*)	= an $N\times K$ matrix of standard deviations,
Gmax(*list*,*index*)	= an $N\times K$ matrix of group maxima,
Gmin(*list*,*index*)	= an $N\times K$ matrix of group minima.

You may use these matrices in data transformations. Generally, this will involve some use of the group means. Suppose your group indicator is named *state* and you have a panel of observations on *N* states. You could transform a variable named, say, *income* to deviations from the specific state means with the following commands:

```
MATRIX    ; stmean = Gxbr(income,state) $
CREATE    ; incdev  = income - stmean(state) $
```

Note how the stratification variable is used as a subscript in the **CREATE** command. The two commands do the following computations

$$\overline{stmean}_{state} = \frac{1}{n_{state}} \sum_{i=1}^{n_{state}} income_{i,state}\ ,\ state = 1,\ldots,NSTATES,$$

$$incdev_{i,state} = income_{i,state} - \overline{stmean}_{state}\ ,\ i = 1,..,n_{state},\ state = 1,\ldots,NSTATES.$$

There is also a built in **CREATE** command that would do this at once,

```
CREATE    ; incdev = Group Devs(income, Str = state) $
```

The matrix function

Gsum(*namelist* [, *variables not in a list*], *weight, index*)

(the additional variables are optional) is a general function that transforms the data set into a matrix of weighted sums. The resulting matrix has number of rows equal to the number of groups in the index set. The number of columns is the number of variables in the namelist and the additional variables listed. The variables in the namelist are weighted by the weighting variable as they are summed while the variables in the list of additional variables, if any are specified, are simply summed, but not weighted.

This function is usually of the form Gsum(*x*,*index*)to compute group sums and the GMM weighting matrix. An alternative form may be used to provide weights for some or all of the variables, using

Gsum(*x*, *v*1, *v*2, ..., *weights*, *index*)

(and likewise for Gmmw described below). In this form, the weights are applied to the individual observations, when summing over the variables in *x*, but not *v*1, *v*2, ...

NOTE: Gxbr, Gsdv, etc. automatically bypass missing values rather than exiting when missing values show up. This is a change from earlier versions of *LIMDEP*.

R16.11.2 GMM Weighting Matrix for Panel Data

A GMM weighting matrix for a panel data application is created with the function

Gmmw(*list, e, index*) **=** a $K{\times}X$ GMM style weighting matrix.

In this function, the variable *e* is typically a residual vector and *list* is a namelist of a set of instrumental variables. This function computes a panel data based weighting matrix of the form

Gmmw(*x, e, index*) $= \Sigma_{i=1}^{N}(\mathbf{X}_i'\mathbf{e}_i)(\mathbf{e}_i'\mathbf{X}_i) = \Sigma_{i=1}^{N}\mathbf{w}_i\mathbf{w}_i'$ where $\mathbf{w}_i = \Sigma_{t=1}^{T_i}\mathbf{x}_{it}e_{it}$

and $\mathbf{x}_{it}$ is the column vector of data on the K variables in $\mathbf{X}_i$ for individual i at time t. Thus, $\mathbf{X}_i$ is a $T_i{\times}K$ matrix of data for individual i. This can be adapted to a cross section application simply by making the index variable define groups of one. For example,

```
CREATE     ; i = Trn(1,1) $
MATRIX     ; w = 1/n * Gmmw(x,e,i) $
```

computes $\mathbf{W} = \frac{1}{N}\Sigma_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i'e_i^2$. This is the usual weighting matrix for GMM estimation with a cross section.

R16.11.3 Gsum and Gmmw Functions with Weights for Some or All Variables

The Gsum and Gmmw functions for panel data are usually of the form

; Gsum(x, index) and ; Gmmw(x, e, index)

to compute group sums and a GMM weighting matrix. An alternative form may be used to provide weights for some or all of the variables, using

Gsum(*x, v1, v2, ..., weights, index*)
and Gmmw(*x, v1, v2, ..., e, index*).

In this form, the weights are applied to the individual observations, when summing over the variables in *x*, but not *v*1, *v*2, ...

R16.11.4 Matrix Forms for Computing Moments for Panel Data

These structures are for balanced or unbalanced panel data sets. In the definitions, **xxx** and **yyy** are namelists of variables or the names if individual variables, *u* is the name of a variable, typically a residual, '**panel**' is either the fixed number of periods or the variable giving the group count as usual for *LIMDEP* model command. Note that this differs from the preceding functions, which use the equivalent of a stratification variable. These are strictly based on group counts, either fixed and explicit, or in the form of a group count variable.

MATRIX ; Result = xxx ' [e , panel] yyy $

This computes the sum of moments, $\Sigma_{i=1}^{N}(\mathbf{X}_i'\mathbf{e}_i)(\mathbf{e}_i'\mathbf{Y}_i)=\Sigma_{i=1}^{N}\mathbf{w}_i\mathbf{v}_i'$. This computation is similar to the Gmmw function discussed above, except here, there may be different sets of variables whereas in Gmmw, **xxx** is the same as **yyy**. The result of the computation is $\Sigma_i\mathbf{X}_i'\mathbf{e}_i\times\,\mathbf{e}_i'\mathbf{Y}_i$ where $\mathbf{X}_i$ is $T_i\times K$ and Y_i is $T_i\times M$. The data in the computation above are untransformed. To use group mean deviations in the same computation, use

MATRIX ; Result = xxx ' [- e , panel] yyy $

This computes $\Sigma_{i=1}^{N}(\mathbf{X}_i'\mathbf{M}_i^0\mathbf{e}_i)(\mathbf{e}_i'\mathbf{M}_i^0\mathbf{Y}_i)=\Sigma_{i=1}^{N}\left(\Sigma_{t=1}^{T_i}(\mathbf{x}_{it}-\overline{\mathbf{x}}_i)(e_{it}-\overline{e}_i)\right)\left(\Sigma_{t=1}^{T_i}(e_{it}-\overline{e}_i)(\mathbf{y}_{it}-\overline{\mathbf{y}}_i)'\right)$.

Note that only the minus sign in front of '**e**' differentiates these two. To do the summing without weighting, we obtain within group moment matrices. Use

MATRIX ; Result = xxx ' [- , panel] yyy $

This computes $\Sigma_{i=1}^{N}(\mathbf{X}_i'\mathbf{M}_i^0\mathbf{Y}_i)=\Sigma_{i=1}^{N}\left(\Sigma_{t=1}^{T_i}(\mathbf{x}_{it}-\overline{\mathbf{x}}_i)(\mathbf{y}_{it}-\overline{\mathbf{y}}_i)'\right)$. Note that there is a minus sign without a variable name after the opening bracket. This form just computes within groups sums of squares and cross products – i.e., moment matrices based on group mean deviations. To compute the between groups sums of squares,

MATRIX ; Result = xxx ' [+ , panel] yyy $

This computes $\Sigma_{i=1}^{N}(\mathbf{X}_i'(\mathbf{I}-\mathbf{M}_i^0)\mathbf{Y}_i)=\Sigma_{i=1}^{N}\left(\Sigma_{t=1}^{T_i}T_i\overline{\mathbf{x}}_i\overline{\mathbf{y}}_i'\right)$, which is the between groups sums of squares and cross products. Note, again, no '**e**' variable is given. Finally, the total sum of squares for the panel is

MATRIX ; Result = xxx ' [* , panel] yyy $

This computes $\Sigma_{i=1}^{N}\left(\Sigma_{t=1}^{T_i}(\mathbf{x}_{it}-\overline{\mathbf{x}})(\mathbf{y}_{it}-\overline{\mathbf{y}})'\right)=\Sigma_{i=1}^{N}\Sigma_{t=1}^{T_i}(\mathbf{x}_{it}-\overline{\mathbf{x}})(\mathbf{y}_{it}-\overline{\mathbf{y}})'$.

NOTE: None of these commands modify the input data. They also bypass missing data.

R17: Using the Calculator

R17.1 Introduction

You will often need to calculate scalar results. The scientific calculator, **CALC** is provided for this purpose. For example, you can use **CALC** to look up critical points for the normal, t, F, and chi squared distributions instead of searching a table for the appropriate value. (And, **CALC** will give you any value, not just the few in the tables.) Another case will be calculation of a test statistic such as a t ratio or likelihood ratio statistic. *LIMDEP*'s calculator can function like a hand calculator, but, it is an integral part of the larger program as well. Results you produce with the calculator can be used elsewhere, and results you obtain elsewhere, such as by computing a regression, can be used in scalar calculations. The programs listed in the chapters to follow contain numerous examples. The example below is a simple application.

Example: Testing for a Common Parameter in a Probit Model

Suppose a sample consists of 1000 observations in 10 groups of 100. The subsamples are observations 1-100, 101-200, etc. We consider a probit model, $y^* = \beta_0+\beta_1 x+\varepsilon$, observed $y = 1$ if $y^* > 0$ and 0 otherwise. With $\varepsilon \sim N[0,1]$, $\text{Prob}[y=1]=\Phi(\beta_0+\beta_1\text{x})$. We are interested in using a likelihood ratio statistic to test the hypothesis that the same parameters, β_0 and β_1, apply to all 10 subsamples against the alternative that the parameters vary across the groups. We also want to examine the set of coefficients.

The first two commands initialize the log likelihood function and define a place to store the estimates.

```
CALC        ; lu = 0 ; i1 = 1 $
MATRIX      ; slopes = Init(10,2,0) $
```

The following commands define a procedure to compute probit models and sum unrestricted logL.

```
PROC $
CALC        ; i2 = i1 + 99 $
SAMPLE      ; i1 - i2 $
PROBIT      ; Lhs = y ; Rhs = one,x $
CALC        ; lu = lu + logl ; i1 = i1 + 100 $
MATRIX      ; slopes(i,*) = b $
ENDPROC $
```

Execute the procedure 10 times, resetting the sample each time.

```
EXECUTE     ; i = 1,10 $
```

The next two commands compute the restricted log likelihood.

```
SAMPLE      ; 1-1000 $
PROBIT      ; Lhs = y ; Rhs = one,x $  Computes restricted (pooled) logL.
```

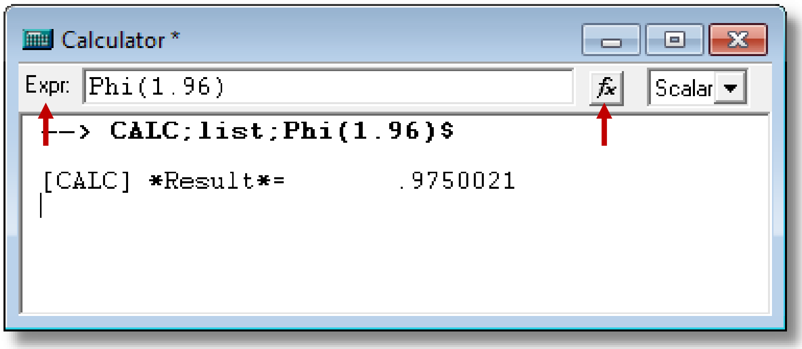
Now, carry out the likelihood ratio test.

```
CALC        ; chisq = 2 * (lu - logl) ; prob = 1 - Chi(chisq,18) $
```

CALC plays several roles in this example. It is used to accumulate the unrestricted log likelihood function, *lu*. The counters *i*1 and *i*2 are set and incremented to set the sample to 1-100, 101-200, etc. The loop index, *i*, is also a calculator scalar, and once it is defined, any other command, such as the **MATRIX** command above, can use *i* like any other number. Finally, the last **CALC** command retrieves the log likelihood from the unrestricted model, computes the test statistic, then computes the tail area to the right of the statistic to determine if the hypothesis should be rejected.

R17.2 Command Input in CALCULATE

CALCULATE is the same as **MATRIX** in the two modes of input. Select Tools:Scalar Calculator to open the calculator window, shown in Figure R17.1.



```
--> CALC;list;Phi(1.96)$

[CALC] *Result*=       .9750021
```

Figure R17.1 Calculator Window

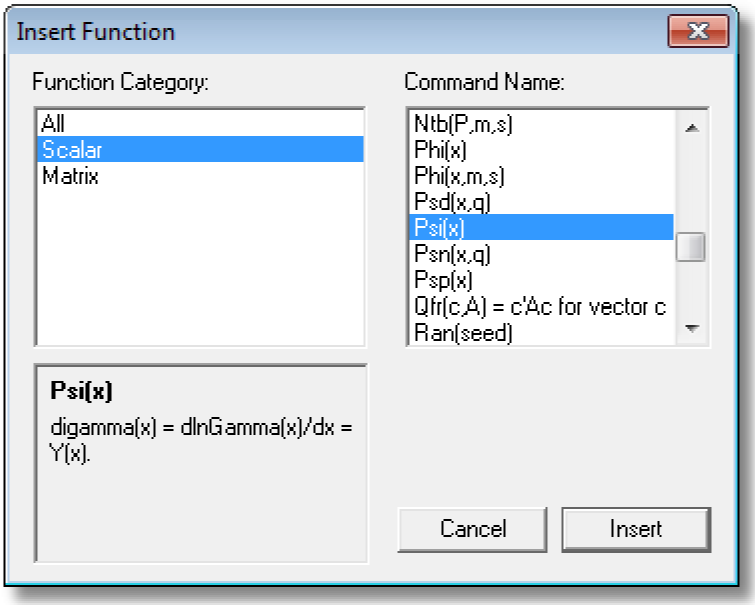
There are two ways to enter commands in the calculator window. You can type **CALCULATE** commands in the smaller 'Expr:' window. If your command will not fit on one line, just keep typing. At some convenient point, the cursor will automatically drop down to the next line. Only press Enter when you are done entering the entire command. In this mode of entry, you do not have to end your commands with a $.

Alternatively, you can click the f_x button to open a subsidiary window, the Insert Function dialog box that provides a menu of matrix and calculator functions (see Figure R17.2). Select Scalar to display the calculator functions (described in Section R17.6). By selecting a function and clicking Insert, you can insert a template for the indicated function into your 'Expr:' window in the calculator window. You must then change the arguments in the function (e.g., the '*x*' in the Phi(*x*) in Figure R17.2) to the entity that you desire. When you have entered your full expression in the window, press Enter to display the command in the lower part of the window, as shown above.

If your command is part of a program, it is more likely that you will enter it in 'command mode' or in what we will label the 'in line' format. You will use this format in the editing window. That is, in the format,

CALC ; ... the desired result ... $

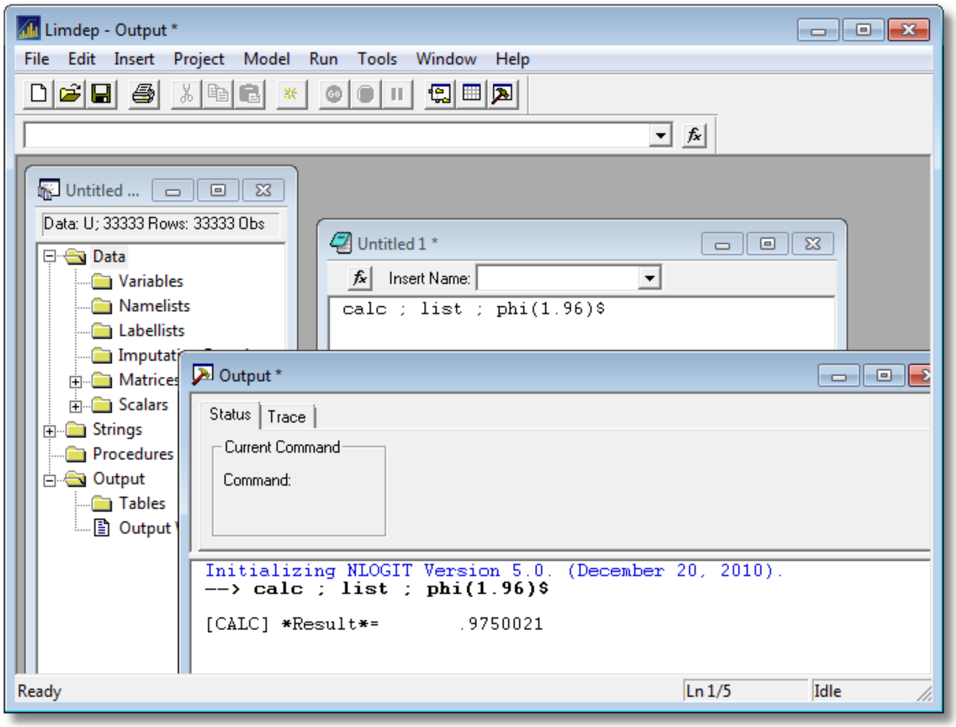
Commands may be entered in this format from the editor, as part of a procedure, or in an input file. See Figure R17.3. One difference between the calculator window and display in the text editor or the output window is that in the latter, you must include **; List** in your command to have the result actually displayed. This is the same as **MATRIX**. See Section R17.3 for details on the **; List** specification.



Psi(x)

digamma(x) = dlnGamma(x)/dx = Y(x).

Figure R17.2 Insert Function Dialog Box for Calculator Functions



```
calc ; list ; phi(1.96)$
```

```
Initializing NLOGIT Version 5.0. (December 20, 2010).
--> calc ; list ; phi(1.96)$

[CALC] *Result*=        .9750021
```

Figure R17.3 CALCULATE Command in Text Editor

CALCULATE is similar to **CREATE** at this point, except that instead of calculating whole columns of data, you calculate single, or ‘scalar’ values. When you give a name to the result, it is kept in a work area and you can use it later. For example, suppose you wanted to have the value of e (Euler’s constant) to use later on. You could, for example, calculate **e = Exp(1)**. You could then

```
CREATE        ; etotheax = e ^ (a*x) $
```

CALC is also similar to **MATRIX** in that if you wish to see a value without keeping it you may type the expression without giving it a name, as in

```
CALC          ; 1 + pi * Log(25) + 2.5 / 1.23 $
```

or, for the 99% critical value for a two tailed test from the standard normal distribution, Ntb(.995). (Ntb stands for Normal table. You also have a t table, and so on.)

R17.3 Results from CALCULATE

As shown above, when you are in the calculator window, the result of a calculator expression, named or not, is displayed on your screen when it is obtained. When **CALC** commands are given in command mode, the default is not to display the results of any computations in the output window or in the output file if one is open. We assume that in this mode, results are intermediate computations, for example, the increments to the counters in the example in Section R17.1. Commands that you give will be listed in your trace file in all cases and in your output window.

You can request a full display of results both in the output window and in an output file by placing

```
              ; List
```

before the result to be listed. You can turn this switch off with

```
              ; Nolist
```

Thus, the command **CALC ; tailprob = Phi(1) $** will create a named scalar, but will not show any visible numerical results. But,

```
CALC          ; List ; tailprob = Phi(1) $
```

will show the result on the screen in the output window. Once the end of a command is reached, **; Nolist** once again becomes the default. The **; Nolist** and **; List** switches may be used to suppress and restore output at any point. When the **; Nolist** specification appears in a **CALC** command, no further output appears until the **; List** specification is used to restore the listing. At the beginning of a command, the **; List** switch is *off*, regardless of where it was before.

To see a result that was computed earlier, there are several ways to proceed. A **CALC** command can simply ‘calculate’ a name. Thus, in the command format, you could just give the command

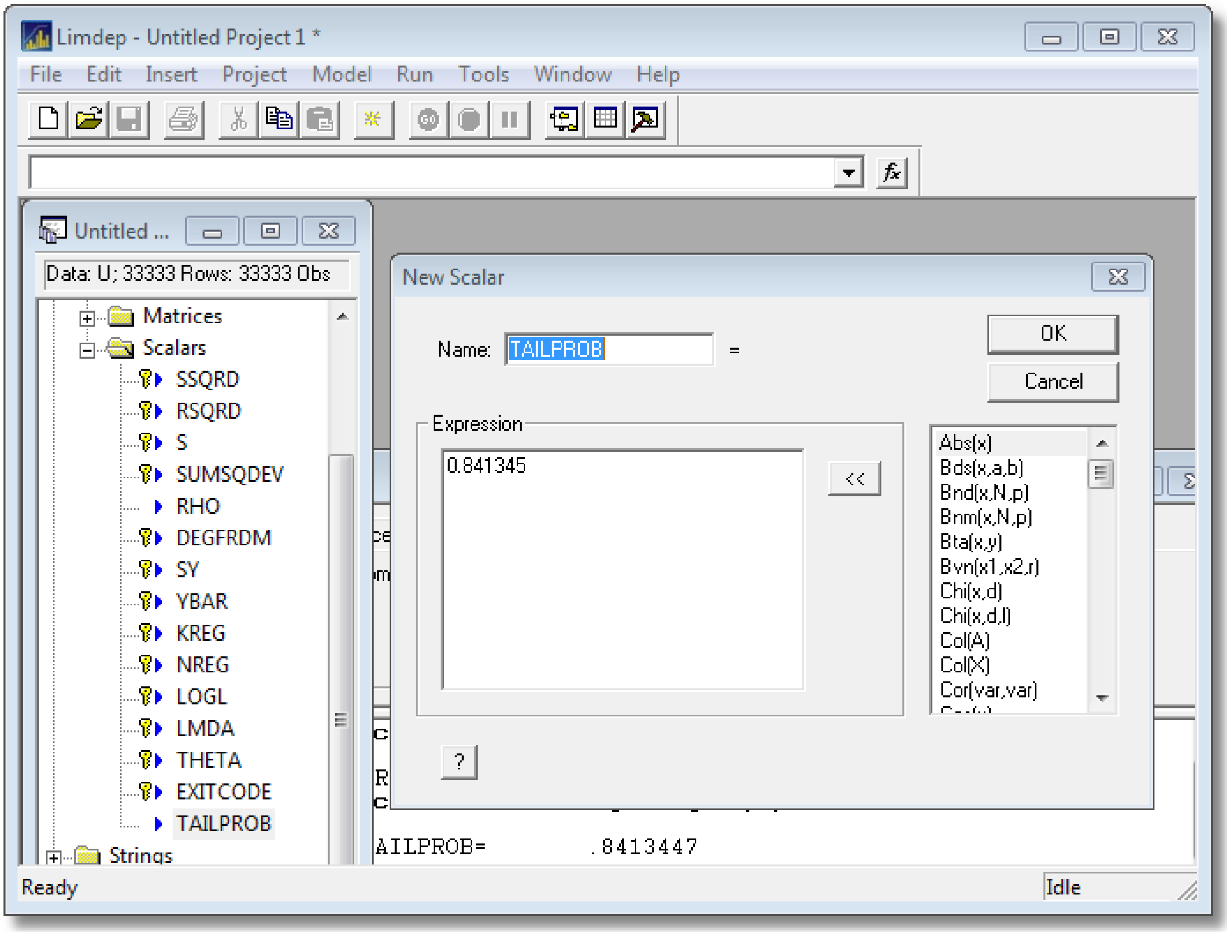
```
CALC          ; List ; tailprob $
```

You may also open the calculator window and just type the name of the scalar you want to see. Finally, when you obtain a named scalar result, it will be added to the project window. (You must 'open' the Scalars data group by clicking the ⊞.) When the list of scalars is displayed, click any name to display the value at the bottom of the window, in the border. Double clicking a scalar name will open the New Scalar dialog box which may also be used to replace the value of that scalar. See Figure R17.4.

As with **MATRIX**, you can see the full internal 17 digit result for your **CALC** commands by using **; Peek** instead of **; List**. For our earlier example, the full value of Phi(1.96) is found by

CALC ; Peek ; Phi(1.96) $

```
[CALC] *Result*=  .97500210485177950D+00
```



Data: U; 33333 Rows: 33333 Obs

Name: TAILPROB =

0.841345

AILPROB=   .8413447

Figure R17.4 Edit Function of New Scalar Dialog Box

R17.4 Forms of CALCULATE Commands – Conditional Commands

The essential format of a **CALCULATE** command is

CALC ; name = result ; ... additional commands ... $

If you wish to see the '**result**' but do not wish to keep it, just omit '**; name =** .' The same applies to the dialog mode in the calculator window. Scalar results will be mixtures of algebraic expressions (addition, multiplication, subtraction, and division), functions, such as logs, probabilities, etc., and, possibly, algebraic manipulation of functions of scalars or expressions.

All calculator commands may be made conditional, in the same manner as **CREATE** or **MATRIX**. The conditional command would normally appear

CALC ; If (logical expression) name = expression $

The logical expression may be any expression that resolves either to 'true' or 'false' or to a numeric value, with nonzero implying true. The rules for the expression are identical to those for **CREATE** (see Section R4.2.2) and **REJECT** (see Section R7.2.2), as well as **MATRIX**, and all forms of **DO**. In this setting, if the condition is true, '*name*' is computed; if it is false, *name* is not computed. Thus, if *name* is a new scalar, and the condition is false, after the command is given, *name* will not exist. For example,

CALC ; If (A(1,1) > rsqrd) q = Log(Dtr(sigma)) $

An entire set of **CALCULATE** commands can be made conditional by placing a semicolon after the condition, as in

CALC ; If (condition) ; name = result ; result $

If the condition is false, none of the commands which follow it are carried out. This form of condition may appear anywhere in a group of **CALC** commands. This will be most useful in iterative programs to condition your **CALC** commands.

R17.4.1 Reserved Names

You can have a total of 100 scalar results stored in your work area. You can obtain a complete list of the names and values assigned to any scalars in the calculator work area by navigating the project window. Fourteen of the scalars are used by the program to save estimation results, and are reserved. The 14 reserved names are

ssqrd, degfrdm, ybar, logl, kreg, sumsqdev, rsqrd, sy, rho, lmda, nreg, theta, s, exitcode

You can see the reserved scalars in the project window in Figure R17.4. They are the ones marked as 'locked' with the 🔒 symbol,. These scalars (save for *rho*– see the hint below) are 'read only.' You may not change them with your commands. Most of these results apply to the linear regression model, but values such as *ybar*, *sy*, and *logl* are saved by nearly all models. Scalars *lmda* and *theta* will change from model to model, depending on the ancillary parameters in the model. After you estimate a model, you will find these scalars defined automatically with the indicated values. These values can thereafter be used on the right hand side of any command. The final one, *exitcode*, is an indicator of the success or failure of the most recent estimation command. Usage is described in Section R26.5.

TIP: When you use **EXECUTE ; name = values $** (see Chapter R19), *name* becomes a 'read only' scalar while the procedure is being executed. After the loop is finished, *name* will still exist, and you can modify it any way you wish. Notice, for instance, in the example in Section R17.1, **CALC** uses the loop index, *i* to obtain *i*1and *i*2 for the sample setting. But, that procedure could not change *i* while it is executing.

HINT: Since it is such a common application, there is an exception to the read only setting of these scalars. The scalar *rho* may be set by a loop control. For example, for scanning in a model of autocorrelation, you might **EXECUTE ; rho = 0, 1, .025 $**. In general *rho* is not a protected name. However, you cannot delete *rho*.

R17.4.2 Work Space for the Calculator

Although there are 100 scalars available, the 14 protected names leave you a total of 86 to work with. If you find yourself running out of room, the command

```
CALC          ; Delete name, name, ... $
```

can be used to clear space. Note that there is no comma or semicolon between the **; Delete** specification and the first scalar name. You may also delete scalars that are not reserved in the project window by highlighting their names and pressing the Del key.

TIP: You are not really limited to these 86 scalars. Any 1×1 matrix can be used as if it were a named scalar, so the distinction disappears. This adds nearly 100 named scalars to your capacity. To use these additional scalars, you must create them as matrices, which you can do as follows: **MATRIX ; name = [0] $** For example,

```
MATRIX        ; newsclr = [0] $
CALC          ; newsclr = Phi(0.234) $
```

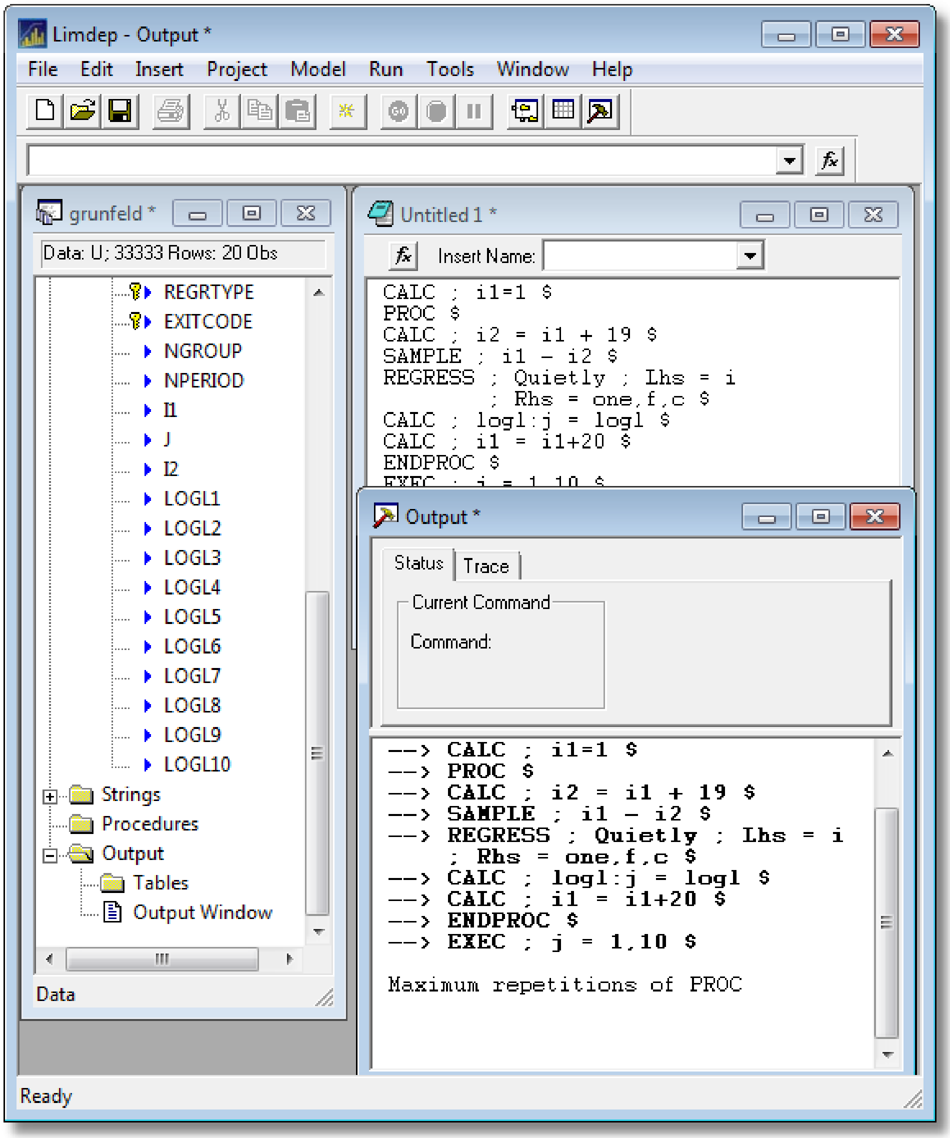
Do note, however, that when you list scalars in the project window, these 1×1 matrices will not be displayed. (They will be displayed as matrices.) But, when you give **CALC** commands, you can use the values taken by these 1×1 matrices just as if they were scalars.

R17.4.3 Compound Names for Scalars

The names of scalars may be indexed by other scalars, in the form *ssss:iiii* where '*ssss*' is a name and '*iiii*' is an integer valued index scalar. For example,

CALC ; i = 37 ; value : i = pi $

creates a scalar named *value*37 and assigns it the value π. The procedure in the editor window in Figure R17.5 shows how one might use this feature. The data set consists of 10 groups of 20 observations. The procedure computes a linear regression model using each subsample. Then it catches the log likelihood function from each regression, and puts it in a correspondingly named scalar. Thus, the loop index, *j*, takes values 1,2,...,10, so the scalar names are *logl:j* = *logl*1,...,*logl*10.



```
CALC ; i1=1 $
PROC $
CALC ; i2 = i1 + 19 $
SAMPLE ; i1 - i2 $
REGRESS ; Quietly ; Lhs = i
        ; Rhs = one,f,c $
CALC ; logl:j = logl $
CALC ; i1 = i1+20 $
ENDPROC $
EXEC ; j = 1,10 $
```

```
--> CALC ; i1=1 $
--> PROC $
--> CALC ; i2 = i1 + 19 $
--> SAMPLE ; i1 - i2 $
--> REGRESS ; Quietly ; Lhs = i
    ; Rhs = one,f,c $
--> CALC ; logl:j = logl $
--> CALC ; i1 = i1+20 $
--> ENDPROC $
--> EXEC ; j = 1,10 $

Maximum repetitions of PROC
```

Figure R17.5 Procedure with Indexed Scalar Names

R17.5 Scalar Expressions

The rules for calculator expressions are identical to those for **CREATE**. The rules of algebra apply, with operations ^ and @ (Box-Cox transformation) taking first precedence, * and / next, followed last by + and -. You may also use any of the functions listed below in any expression. This includes the percentage points or critical values from the normal, t, F, and chi squared distributions, sums of sample values, determinants of matrices, or any other algebraic functions. Chapter R16 describes how to obtain matrix results. You may also use an element of a matrix with its subscript enclosed in parentheses in any scalar calculation. Finally, any particular observation on any variable in your data area may also be used in an expression. For example, you might

```
CREATE     ; x = some function $
CALC       ; q = x(21) * sigma(2,2) $
```

In evaluating subscripts for variables, the observation refers to rows in the data array, not the current sample. Expressions may also contain any number of functions, other operators, numbers, and matrix elements. A scalar may appear on both sides of the equals sign, with the result being replacement of the original value. For examples:

```
CALC ; varsum = b(1)^2 * varb(1,1) + b(2)^2 * varb(2,2) + 2 * b(1) * b(2) * varb(1,2) $
CALC ; messy = messy^2/pi - Gma(.5)/Gma(.1) * Sum(age)  $
```

If it is necessary to change the algebraic order of evaluation, or to group parts of an expression, use parentheses nested to as many levels as needed. For example,

```
CALC       ; func = (Gma(3) + Gma(5))^3 + ((x + y)/c) * (f + g) $
```

You may also nest functions. For example,

```
CALC       ; q = Log(Phi(a1 + a2 * Exp(a3 + a4 * Gma(z)))) $
```

There are two constants which can be used by name without having been set by you. At all points in the program, the name '*pi*' will be understood to be the number π = 3.14159... Note that this will preempt matrices and scalars named *pi*, so this name should be avoided in other contexts. The name *pi* may also appear in **MATRIX** and **CREATE** commands, for example,

```
MATRIX     ; pii = pi * Iden(5) $
CREATE     ; f = 1/(sg * Sqr(2 * pi)) * Exp(-.5 * ((x - mu)/sg)^2) $
```

When you give a **CALC**, **MATRIX**, or **CREATE** command, the name '*n*' is always taken to mean the current sample size. You may use *n* in any scalar calculation. For example, after you compute a regression, the log likelihood function could be computed using

```
CALC       ; l = -n/2 * (1 + Log(2 * pi) + Log(sumsqdev/n)) $
```

NOTE: *n* and *pi* have the meanings described above everywhere in *LIMDEP*. Thus, you could use *pi* in a list of starting values, as part of a model command, or in **CALCULATE**.

R17.6 Calculator Functions

The functions listed below may appear anywhere in any expression. The arguments of the functions can be any number within the range of the function (e.g., you cannot take the square root of -1) as well as matrix elements and names of other scalars. Function arguments may also be expressions, or other functions whose arguments may, in turn, be expressions or other functions, and so on. For example,

z = Log(Phi(a1 + a2 + Log(a2 + (q + r)^2)))

is a valid expression which could appear in a **CALC** command. The depth of nesting functions allowed is essentially unlimited. When in doubt about the order of evaluation, you should add parentheses to remove the ambiguity. Also, in functions which have more than one argument separated by commas, such as Eql(*x*,*y*) (which equals one if *x* equals *y*), include expression(s) in parentheses. For example,

Eql(*x*+*y* , (*r*+*c*)^2)

may not evaluate correctly because of the '*x*+*y*' term. But,

Eql((*x*+*y*) , ((*r*+*c*)^2))

will be fine. The supported functions are listed below:

R17.6.1 Basic Algebraic and Trigonometric Functions

Log(*x*)	= natural log
Abs(*x*)	= absolute value
Sin(*x*)	= sine
Tan(*x*)	= tangent
Exp(*x*)	= exponent
Sqr(*x*)	= square root
Cos(*x*)	= cosine
Rcs(*x*)	= arccosine
Rsn(*x*)	= arcsine
Ath(*x*)	= hyperbolic arctangent = ½ ln[(1+*x*)/(1-*x*)]
Ati(*x*)	= inverse hyperbolic arctangent = [exp(2*x*)-1] / [exp(2*x*)+1]
Atn(*x*)	= arctangent
Ash(*x*)	= hyperbolic arcsinc = Log(*x* + Sqr(1+x^2))
As1(*x*)	= dAsh(*x*)/d*x* = 1 / Sqr(1+x^2)
Ach(*x*)	= hyperbolic arccosine = Log(*x*+Sqr(x^2-1))
Ac1(*x*)	= dAch(*x*)/d*x* = 1/Sqr(x^2 – 1)
Ath(*x*)	= hyperbolic arctangent = ½Log((1+*x*)/(1-*x*)), -1 <*x*< 1
At1(*x*)	= 1/ (1 – x^2)
Ati(*x*)	= inverse hyperbolic arctangent = [Exp(2*x*)-1]/[Exp(2*x*)+1]
Hsn(*x*)	= hyperbolic sin = .5 (Exp(2*x*)-1)/Exp(*x*)
Hs1(*x*)	= dHsn(*x*)/d*x* = Hcs(*x*)
Hcs(*x*)	= hyperbolic cos = .5*(Exp(2*x*)+1)/Exp(*x*)
Hc1(*x*)	= dHcs(*x*)/d*x* = Hsn(*x*)
Htn(*x*)	= hyperbolic tangent = Hsn(*x*)/Hcs(*x*)
Ht1(*x*)	= dHtn(*x*)/d*x*) = 1 / Hcs(*x*)^2

R17.6.2 Relational Functions

Eql(x,y)	= 1 if x equals y, 0 if not
Neq(x,y)	= 1 - Eql(x,y)
Sgn(x)	= 1 if $x > 0$, 0 if $x = 0$, -1 if $x < 0$

R17.6.3 Critical Points from the Normal Family of Distributions

In each case, when you enter 'Fcn(P,...)' where P is the probability, *LIMDEP* finds the x such that for that distribution, the probability that the variable is less than or equal to x is P. For example, for the normal distribution, Ntb(.95) = 1.645. The P you give must be strictly between 0 and 1.

Ntb(P)	= standard normal distribution
Inp(P)	= same as Ntb(P)
Ttb(P,d)	= t distribution with d degrees of freedom
Ctb(P,d)	= chi squared with d degrees of freedom
Ftb(P,n,d)	= F with n numerator and d denominator degrees of freedom
Ntb(P,μ,σ)	= normal distribution with mean μ and standard deviation σ
Erf(x)	= error function = $2\Phi(\text{Sqr}(2)x) - 1$
Erc(x)	= complementary error function = $2(1 - \Phi(\text{Sqr}(2)x))$

R17.6.4 Probabilities and Densities for Continuous Distributions

Phi(x)	= probability that N[0,1] $\le x$
Phi(x,μ,σ)	= probability that N[μ,σ] $\le x$
N01(x)	= density of the standard normal evaluated at x (Note 'N-zero-one')
Lgf(x)	= log of standard normal density = $-½ (\ln 2\pi + x^2)$. Lgf(0)=.918938542
N01(x,μ,σ)	= density of normal[μ,σ] evaluated at x
Tds(x,d)	= prob[t with d degrees of freedom $\le x$]
Chi(x,d)	= prob[chi squared variable with d degrees of freedom $\le x$]
Fds(x,n,d)	= prob[F with n numerator and d denominator degrees of freedom $\le x$]
Lgp(x)	= logit probability = exp(x)/(1+exp(x))
Lgd(x)	= logit density = Lgp(x)×(1 - Lgp(x))
Lgt(P)	= logit of x = Log(P/(1-P)) for $0 < P < 1$
Xpn(x,θ)	= prob[exponential variable with mean $1/\theta \le x$]
Bds(x,α,β)	= prob[beta variable with parameters $\alpha,\beta \le x$]
Bdd(x,α,β)	= density function for beta variable x with parameters α,β
Kp1(df)	= 1% critical value for Kodde-Palm mixture of χ^2[0] and χ^2[df]
Kp5(df)	= 5% critical value for Kodde-Palm mixture of χ^2[0] and χ^2[df]
Tdd(x,d)	= density of t with d degrees of freedom
Tdf(x,d)	= CDF of t with d degrees of freedom
Cdd(x,d)	= density of chi squared with d degrees of freedom
Cdf(x,d)	= CDF of chi squared with d degrees of freedom
Fcs(x,σ_1,σ_2)	= density for closed skew normal
Pcs(x,σ_1,σ_2)	= CDF for closed skew normal

R17.6.5 Moments of the Left Truncated Normal Distribution

Trm(*a*)	= mean of the truncated normal distribution, left truncated at *a*
Trv(*a*)	= variance of the left truncated at *a* normal distribution
Trm(*a*)	= E[z|$z \geq a$] = $\phi(a)/\Phi(-a)$, standard normal distribution
Trv(*a*)	= Var[z|$z \geq$a] = 1 - Trm(*a*)(*a* + Trm(*a*)), standard normal
Trm(*a*,μ,σ)	= μ+ σTrm((*a*-μ)/σ)
Trv(*a*,μ,σ)	= σ^2Trv((*a*-μ)/σ)

These last two change the mean and standard deviation from 0 to μ and 1 to σ, respectively. For upper (right) truncation instead of lower, add a ‘1’ as the final argument.

Trm(*a*,1), Trm(*a*,μ,σ,1), Trv(*a*,1), Trv(*a*,μ,σ,1).

R17.6.6 Probabilities and Densities for the Bivariate Normal Distribution

Bvn(*x*1,*x*2,*r*)	= cumulative probability from the bivariate standard normal distribution,
Bvd(*x*1,*x*2,*r*)	= density from the bivariate standard normal,
Bv1(*x*1,*x*2,*r*)	= partial derivative of Bvn with respect to *x*1 (see Greene, 2012, p. 740),
Bv2(*x*1,*x*2,*r*)	= partial derivative of Bvn with respect to *x*2.

R17.6.7 Probabilities and Densities for the Multivariate Normal Distribution

In the following, *x* must be a vector with *M* elements that was created with a **MATRIX** command or as a byproduct of some estimation program. *W* must be a square *M*×*M* covariance matrix for the distribution. Then,

Mvn(*x*,*W*)	= multivariate normal CDF with mean vector zero, Prob[$X \leq\leq x$],
Mvd(*x*,*W*)	= multivariate normal PDF.

The multivariate CDF is

$$F(\mathrm{x}) = \int_{-\infty}^{B_M}\int_{-\infty}^{B_{M-1}}\cdots\int_{-\infty}^{B_1} (2\pi 2^{-M/2} \,|\, \mathbf{W} \,|^{-1/2} \exp[(-1/2)\mathbf{x}'\mathbf{W}^{-1}\mathbf{x}]dx_1\cdots dx_{M-1}dx_M \,.$$

The lower limits are all A_m = -∞. Thus, **x**(.) provides the upper bounds, $B_1,...,B_M$. For instance, one of the examples in Breslaw’s contribution to the (1994) *ReStat* symposium – you can use this to test the computation – is

```
MATRIX    ; x = [0/0/0/0] ; w = [1/.2,1/.2,.4,1/.2,.4,.6,1] $
CALC      ; p4 = Mvn(x,w) $
```

which will produce a value close to 0.15. If you desire to compute the probability in a rectangle defined by finite lower bounds, $A_1,...,A_M$, at the lower limits and $x_1,...,x_M$ at the upper limits, use

```
CALC      ; Result = Mvn(x,w,a) $
```

If you desire complementary probabilities, that is the probability for the area defined by a lower bound of *x*(.) and upper bounds of +∞, use Mvn(*y*,*W*) where *y* is the negative of *x*. If you desire some of the *x*s to be lower bounds and others to be upper bounds, you can use the following trick: Create an $M \times M$ matrix *T* in which all off diagonal elements are zero and diagonal element T_i is +1 if *x*(*i*) is an upper bound and -1 if T_i is a lower bound. Then, instead of *x* and *W* in your Mvn function, use

xa = Tx and **WA = TWT**

You must create these with **MATRIX** before using the Mvn function in **CALC**.

Multivariate normal probabilities are computed using the GHK simulator (see Section R26.8). The number of replications used for the simulation is set by default at 100 when *LIMDEP* starts up. If you wish to use some other value, use the function

Rep(*Nrep*)	= number of replications for Mvn functions.

R17.6.8 Probabilities for Noncentral Distributions

The right hand tail probabilities for the noncentral chi squared and (singly – numerator only) noncentral F distributions may be obtained with **CALC** by adding the noncentrality parameter to the list for the corresponding central distribution.

Chi(*x*,*d*,*q*)	= prob[noncentral chi square with *d* degrees of freedom and noncentrality parameter *q* is ≤*x*],
Fds(*x*,*n*,*d*,*q*)	= prob[noncentral F with *n* numerator degrees of freedom, *d* denominator degrees of freedom and noncentrality parameter *q* is ≤*x*].

R17.6.9 Probabilities for Discrete Distributions

Psn(*x*,λ)	= prob[Poisson with parameter λ≤*x*]
Psd(*x*,λ)	= prob[Poisson with parameter λ equals *x*]
Bnm(*x*,*n*,π)	= prob[binomial; *n* trials, success probability π≤*x*]
Bnd(*x*,*n*,π)	= prob[binomial; *n* trials, success probability π equals *x*]
Gep(*x*,π)	= prob[geometric; success probability π≤*x*]
Geo(*x*,π)	= prob[geometric; success probability π equals *x*]
Fnb(x,λ,q)	= prob[negative binomial with mean λ and scale *q* equals *x*]
Pnb(x,λ,q)	= prob[negative binomial with mean λ and scale *q* ≤ *x*]

R17.6.10 Gamma Function and Gamma Distribution

Gma(*x*)	= gamma function. Gma(.5) = √π and = (*x*-1)! if *x* = integer
Psi(*x*)	= digamma(*x*) = dlogGamma(*x*)/d*x*
Psp(*x*)	= trigamma(*x*) = $d^2\log\Gamma(x)/dx^2$ = Psi'(*x*) (0 ≤*x* ≤ 40)
Lgm(*x*)	= log of Gma(*x*). Note: Lgm(*x*+1) = log(*x*!)
Bta(*x*,*y*)	= beta function. Bta(*x*,*y*) = Γ(*x*)Γ(*y*)/Γ(*x*+*y*)

R17.6.11 The Incomplete Gamma Function

The gamma density is $f(x) = (a^P / \Gamma(P))e^{-ax} x^{P-1}$, $x \geq 0$, a, $P>0$.

Gmp(x,P,a)	= cumulative probability = Prob[$X \leq x$]
Gmp(x,P,1)	= Gmp(x,P). If a = 1, it may be omitted from the gamma probability.
Gtb(*prob*,P,a)	= inverse probability function, i.e., the x such that the CDF at x equals the probability. As before, if a = 1, it may be omitted.

The normalized *incomplete* gamma integral is

$$g(x,P) = \frac{1}{\Gamma(P)}\int_0^x e^{-at} t^{P-1} dt = g(x,P), \lim_{x\to\infty} g(x,P) = 1.$$

This is the probability given above. Thus, the integral, itself can be computed with $\Gamma(P)g(x,P)$ = Gma(P) * Gmp(x,P). The counterpart for the nonstandardized distribution is obtained by providing a value for a.

R17.6.12 Random Numbers

Rnn(μ,σ)	= one draw from the normal distribution
Rnu(*lower*,*upper*)	= one draw from the continuous uniform distribution
Ran(*seed*)	= sets the seed for the random number generator

If you wish to replicate a set of random draws, set the seed before drawing the sample. By default the seed is set by the system clock, so samples will not be replicated unless you do this. Use an odd number for the seed.

R17.6.13 Matrix Dimensions and Functions

If A is the name of a matrix,

Row(A)	= number of rows in matrix A
Col(A)	= number of columns in matrix A
Rnk(A)	= rank of matrix A
Nrm(A)	= norm of A = trace($A'A$)
2nr(A)	= 2-norm of A = largest singular value

For square matrix A,

Trc(A)	= trace of matrix A
Det(A)	= determinant of matrix A
Lmd(A)	= log of determinant of matrix A if A is positive definite
Cnd(A)	= condition number for matrix A
2nr(A)	= 2-norm of c = largest singular value
2cn(A)	= singular value rank = largest singular value/smallest (if positive)

If 'X' is the name of a namelist, then

Row(X)	= number of observations in current sample = n
Col(X)	= number of variables in the namelist

R17.6.14 Sample Statistics and Regression Results

The observations used in any of the following are the current sample less any missing observations. For the Sum, Xbr, Var, and Sdv functions of a single variable, missing data are checked for the particular variable. Thus, Xbr($x1$) and Xbr($x2$) may be based on different observations. You should keep close track of this if your data have gaps or different sample lengths. For the remaining functions, all observations are used without regard to missing data. For example, in the covariance function, *LIMDEP* uses all data points, so some data may be missing. Be careful using these to prevent the -999s from distorting the statistics. Computations are based on the current sample. You can change the sample for purposes of the calculations with

CALC ; For [variable = value] ; the rest of the command $

For example, to restrict attention to a particular year of the sample, you might use

CALC ; For [year = 2015] ; … $

Type of Data

It might be necessary to base a calculation on the type of data involved. Distinguishing between discrete and continuous data might be a case. The function

Typ(*variable*)	= type of data

is provided for this purpose. The Typ(variable) function returns values

0 = type cannot be determined
1 = fraction limited within zero to one
2 = binary
3 = integers
4 = continuous

Sample Moments

For any variable in your data area, or namelist which contains only one variable name, the functions listed below can be used just like any other function, such as Sqr(2). If you wish only to display the statistic, just calculate it. Otherwise, these functions can be included in any expression.

Sum(*variable*)	= sum of sample values
Xbr(*variable*)	= mean of sample values
Sdv(*variable*)	= standard deviation of sample values
Var(*variable*)	= variance of sample values
Xgm(*variable*)	= the geometric mean; Xgm(x) = Exp[1/nΣ_i log(x_i)]
Xhm(*variable*,*h*)	= the harmonic mean using parameter h; Xhm(x,h) = $[\Sigma_i x_i^h]^{1/h}$
Sku(*variable*)	= skewness coefficient
Krt(*variable*)	= kurtosis coefficient
Rb1(*variable*)	= Sku(*variable*)/Sdv^3(*variable*) = standardized skewness
Rb2(*variable*)	= Krt(*variable*)/Sdv^4(*variable*) - 3 = standardized kurtosis

The summing functions (Sum, Var, Sdv, Xbr) can be restricted to a subsample by including a second variable in the list. If a second variable appears, the function is compute for nonzero values of that second variable. Thus, Sum(*variable*, *dummy*) is the sum of observations for which the dummy variable is nonzero. This allows a simple way to obtain a mean or variance in a subset of the current sample.

Covariance and Correlation

For any pair of variables,

Cov(*variable*,*variable*) = sample covariance
Cor(*variable*,*variable*) = sample correlation

We note, for obtaining the correlation between a continuous variable, *x*, and a binary variable, *d*, one would use the 'biserial' correlation. It turns out that the biserial correlation is equal to the ordinary, Pearson product moment correlation. So no special function is created for this. Just use

CALC ; List ; Cor(*continuous variable x, binary variable d*) **$**

to obtain a biserial correlation coefficient. To request more detailed output for the biserial correlation of a continuous variable and a binary variable, use

Bsr(*continuous variable*,*binary variable*) = biserial correlation.

This produces a detailed set of results. For example, here is the biserial correlation between experience and Gender in the Cornwell and Rupert data using **CALC ; Bsr(exp,fem) $**

```
---------------------------------------------
Biserial correlation of EXP      and FEM
Estimated correlation coefficient = -.09223
Estimated standard error for bsr  =  .01543
Estimated 95%conf. interval=(-.1225,-.0620)
---------------------------------------------
```

Kendall's tau is

$$\tau_{xy} = \frac{\Sigma_{i=1}^{n}\Sigma_{j=1}^{i-1}\operatorname{sgn}(x_i, x_j)\operatorname{sgn}(y_i, y_j)}{n(n-1)/2}$$ where Sgn(x_i,x_j) = +1 if $x_i > x_j$ and -1 if $x_i \leq x_j$.

Compute this with

Ktr(*variable*,*variable*) = Kendall's tau.

Order Statistics

Med(*variable*)	= median of sample values
Min(*variable*)	= sample minimum
Max(*variable*)	= sample maximum
Qnt(*quantile*,*variable*)	= the indicated quantile for the variable

To locate the minimum or maximum value in the current sample, use

Rmn(*variable*)	= observation number where minimum value of variable occurs
Rmx(*variable*)	= observation number where maximum value of variable occurs

Nonparametrics

Rkc(*variable*1,*variable*2) = the rank correlation of two variables

$\rho = 1 - 6\,\Sigma_i\, d_i^2\,/n(n^2 - 1)$, $d_i = variable1_i - variable2_i$

Cnc(*x1*,...,*xK*) = Kendall's coefficient of concordance of the K sets of rankings

$W = 12\Sigma_i(S_i - \overline{S}\,)^2/[nK^2(n^2 - 1)]$ where $S_i = \Sigma_k x_{k,i}$

The coefficient of concordance is used to measure the degree of agreement among n individuals each of which has a set of K ranks. For example, consider a panel of n judges, each ranking a panel of K = 10 paintings or musicians. A large sample chi squared test of the null hypothesis that all n individuals are in 'concordance' may be based on

$$\chi^2[K(n\text{-}1)] \;=\; K(n\text{-}1)W.$$

Dot Products

For any vector (matrix with one row or column), which we denote c or d, or variable in your data set, denoted x or y,

Dot(c,c)	= $c'c$
Dot(c,d)	= $c'd$
Qfr(c,A)	= $c'Ac$ (A is a square matrix conformable with c.)

Two forms of the Dot function are

Dot(x,x)	= $x'x$
Dot(x,y)	= $x'y$

You may also use the simpler form with the apostrophe, and may mix variables and vectors in the function. Thus, if x and y are variables, and c and d are vectors, all of the following are admissible (assuming they are conformable):

CALC ; x'y ; c'y ; Dot(x,y) ; d'd $ and so on.

Regression Statistics

The **CALC** command has several functions which allow you to obtain certain regression statistics, such as an R^2 in isolation from the rest of the least squares computations. In the following, the list of variables in the parentheses is of the form

list = independent variables, dependent variable.

The dependent variable is always given last in a list. As always, if you want a constant term, include *one*. You can use a namelist for the independent variables if you wish, and the wildcard character, *, may be used to abbreviate lists of variable names.

The following functions can be computed, where X is the list of independent variables:

Rsq(X,y)	= R^2 in regression of variable y on X, $R^2 = 1\text{-}e'e/\Sigma_i(y_i - \bar{y})^2$
Xss(X,y)	= explained sum of squares
Ess(X,y)	= error, or residual sum of squares
Tss(X,y)	= total sum of squares
Ser(X,y)	= standard error of regression
Lik(X,y)	= log likelihood function

Count for a Panel

The number of groups in a panel defined by the stratification variable ‘y’ is given by

Ngi(y)	= number of sequences of consecutive identical values of variable y.

This examines the sample of values and counts the number of runs of the same value, assuming that each run defines a stratum. In a sample of 10, if i = 1,1,1,2,2,3,4,4,4, the number of runs (groups) is four.

Pnl(*pds variable*) = average group size for panel defined by pds variable.

R17.7 Fit Measures for a Binary Choice Model

There are a variety of fit measures for binary choice models. (These are discussed in more detail in Chapter E26). For any binary variable, y, and variable p containing a column of fitted probabilities, the function

Fit(y,p)	= table of fit measures for p as a model for predicting y.

The results for this function appear as in the following example:

```
CALC          ; List ; Fit(mode, pfit) $
```

```
+---------------------------------------+
| Fit Measures for Binomial Choice Model |
| Observed = CHOICE   Fitted = CHOICEP   |
+---------------------------------------+
|                Y=0       Y=1     Total|
| Proportions  .75000    .25000   1.00000|
| Sample Size    9600      3200     12800|
+---------------------------------------+
| Log Likelihood Functions for BC Model  |
|             P=0.50    P=N1/N   P=Model|
| LogL =     -8872.28  -7197.89  -6886.72|
+---------------------------------------+
| Fit Measures based on Log Likelihood   |
| McFadden = 1-(L/L0)          =   .04323|
| Estrella = 1-(L/L0)^(-2L0/n) =   .04849|
| R-squared (ML)               =   .04746|
+---------------------------------------+
| Fit Measures Based on Model Predictions|
| Efron                        =   .04339|
| Ben Akiva and Lerman         =   .64204|
| Veall and Zimmerman          =   .08759|
| Cramer                       =   .04509|
+---------------------------------------+
```

The values reported are

$logL$	=	log likelihood $= \Sigma_i y_i \log p_i + (1 - y_i)\log(1 - p_i)$
P_0	=	$(1/N)\Sigma_i\,(1 - y_i)$
P_1	=	$(1/N)\Sigma_i y_i$
N_0	=	$\Sigma_\mathrm{i}\,(1 - y_i)$
N_1	=	$\Sigma_i y_i$
$logL0$	=	restricted (constant term only) log likelihood
	=	$N_0 \log P_0 + N_1 \log P_1$
N	=	$N_0 + N_1$
Efron	=	$1 - [\Sigma_i(y_i - p_i)^2]/[\Sigma_i(y_i - P_1)^2]$ (Note, P_1 = the mean of y.)
McFadden	=	$1 - logL\,/\,logL0$
Ben-Akiva/Lerman	=	$(1/N)\Sigma_i y_i p_i + (1\text{-}y_i)(1 - p_i)$
Cramer	=	$(1/N_1)\Sigma_i\, y_i p_i$ - $(1/N_0)\Sigma_i\, p_i(1 - y_i)$
Veall and Zimmermann	=	$[(\delta - 1)/(\delta - \text{McFadden})] \times \text{McFadden}$, $\delta = N/(2logL0$
R_{ML}^2	=	$1 - \exp[(-2/N)(logL - logL0)]$

R17.8 Hypothesis Tests

Kolmogorov Smirnov Test of Normality

The Kolmogorov-Smirnov test is a nonparametric statistic used to test a distributional assumption. For the implementation here, we use the normal distribution as the null hypothesis. The statistic is computed as

$$D = \max_{1 \le i \le N} \left(F(x_i) - \frac{i-1}{N}, \frac{i}{N} - F(x_i) \right)$$

Where F is the theoretical CDF being tested (normal). For the specified test,

Kst(variable) = Kolmogorov-Smirnov test statistic

The null distribution is assumed to be the normal distribution. The mean and standard deviation of the normal distribution are estimated from the data. The derivation of the behavior of the test statistic, and the critical values, actually assume that the mean and variance of the distribution are known, not estimated from the data. So, the critical values given below should be viewed as approximate. If you do know the mean and standard deviation of the distribution, use

Kst (variable, μ, σ) = Kolmogorov-Smirnov test against N[μ,σ^2].

Critical values of the distribution of the test statistic are as follows:

Sample Size	20	25	30	35	Over 35
95%	.294	.270	.240	.230	1.36/Sqr(*N*)
99%	.356	.320	.290	.270	1.63/Sqr(*N*)

Testing for Significant Differences Between Two Populations

The function Tst(*x*,*y*) is used to test for equality of means or variances for two variables *x* and *y*. The sample used is the current sample. The function is

CALC ; Tst (x,y) $ for the means test,
and **CALC ; Tst (x,y,2) $** for the variance test.

(You may use Tst(*x*,*y*,1) for the means test.) The names *x* and *y* may be two variables, two matrices with any number of rows and columns, or one of each, both of which may be any configuration. Missing values are automatically bypassed, and the samples may be different sizes, as noted. The following shows an application using our discrete choice data.

CALC ; Tst (invc,gc) $

```
Test of equality of    Means.
F statistic with [    1, 1471] =    999.035; P =   .00000
```

CALC ; Tst (invc,gc,2) $

```
Test of equality of Variances.
F statistic with [  839,   839] =      2.197; P =   .00000
```

Testing Equality of the Means of Two Populations

The means test is requested with

```
      CALC      ; Tst (x,y) $
or    CALC      ; Tst (x,y,1) $
```

The test is based on the standard t statistic, which we square to obtain an F statistic with 1 and D degrees of freedom:

$$F[1,d] = (\bar{x} - \bar{y})^2 / (s_x^2 / N_x + s_y^2 / N_y)$$

where $\bar{x} = \dfrac{\Sigma_{i=1}^{N_x} x_i}{N_x}$ and $s_x^2 = \dfrac{\Sigma_{i=1}^{N_x}(x_i - \bar{x})^2}{N_x - 1}$

and likewise for *y*. The sample sizes may be unequal. The inequality of the sample sizes will result if one or the other of the variables contains missing values. As such, for the degrees of freedom for the denominator, we use the Satterthwaite approximation,

$$d = \frac{(s_x^2 / N_x + s_y^2 / N_y)^2}{[(s_x^2 / N_x)^2 / (N_x - 1)] + [(s_y^2 / N_y)^2 / (N_y - 1)]}.$$

Testing Equality of the Variances of Two Populations

The variance equality hypothesis is tested with the command

```
      CALC      ; Tst (x,y,2) $
```

The test statistic is the standard F ratio with N_x - 1 and N_y - 1 degrees of freedom,

$$F[N_x - 1 , N_y - 1] = s_x^2 / s_y^2 .$$

The roles of *x* and *y* are reversed if *y* has the larger variance. (This is merely for convenience in using the *F* table. It has no bearing on the result of the test.)

Testing Equality of Two Population Proportions

This test is requested with the command

```
      CALC      ; Tst (x,y,3) $
```

As before, the data may be provided as variables, matrices or vectors, or a mix of the two. The test is based on the underlying model, Prob[$event_x$] = π_x and Prob[$event_y$] = π_y. The subscripts denote the two populations, not different events. We are interested in testing the null hypothesis

$$H_0: \pi_x = \pi_y$$

based on observed samples, *x* and *y*. These two variables are binary variables indicating the event has occurred ($x_i = 1$ or $y_i = 1$) or not occurred, ($x_i = 0$ or $y_i = 0$). The test statistic is

$$c^2 = \frac{\left(p_x - p_y\right)^2}{P(1-P)\left(1/N_x + 1/N_y\right)} \text{ where } p_j = \frac{\Sigma_{i=1}^{N_j} j_i}{N_j},\ j = x, y$$

and P is the pooled proportion, $(N_x p_x + N_y p_y)/(N_x + N_y)$. Under the null hypothesis of equality, c^2 has a limiting chi squared distribution with one degree of freedom.

Hotelling's T Squared Test of Equality of Means

The statistic is computed for a pair of namelists or a pair of variables. The central result is a Wald type statistic,

$$T^2 = \left(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2\right)' [\text{Covariance Matrix}]^{-1} \left(\overline{\mathbf{x}}_1 - \overline{\mathbf{x}}_2\right).$$

The null hypothesis is that the two means or sets of means are the same. The difference between applications relates to the mean vectors. You can compare the mean of one variable to the mean of another (two variables), the set of means of one set of variables to the means of another set of variables (two lists of variables), or the mean of one group of individuals to another (one list of variables and a dummy variable) or (one variable and one binary variable). For example, the following compares the weeks worked, experience and log wages of union members vs. union members:

```
NAMELIST    ; x = wks,exp,lwage $
CALC        ; Ht2(x,union) $
```

```
------------------------------------------------------
Hotelling t^2 test of equal means X        by UNION
------------------------------------------------------
UNION   =0 Nx =    2649      UNION   =1 Ny =      1516
T squared statistic =   115.1983        K  =         3
F [  3,  4161]      =    38.3810      P value =  .0000
------------------------------------------------------
                Mean         Std.Dev.        Std.Error
WKS     ------------------------------------------
UNION    = 0   47.41223       4.68558           .09104
UNION    = 1   45.76187       5.67451           .14574
EXP     ------------------------------------------
UNION    = 0   19.36316      10.98221           .21338
UNION    = 1   20.71108      10.88929           .27967
LWAGE   ------------------------------------------
UNION    = 0    6.67331        .52286           .01016
UNION    = 1    6.68165        .32793           .00842
------------------------------------------------------
Variances and covariances are assumed to be equal
DF = [K,Nx+Ny-K-1]. F = {(Nx+Ny-K-1)/[K(Nx+Ny-2)]}*T^2
------------------------------------------------------
[CALC]          =      115.1983181
```

Based on the results shown, we would reject the hypothesis of equal means of union members and nonunion workers.

Testing Based on Summary Statistics

The test statistics and procedures described above require you to provide the raw data for computation of the statistics. If you have only the summary statistics, such as the means and variances, you can easily use **CALC** to compute the same test statistics. Three functions are provided to simplify the calculation for you. The three functions are, respectively,

Eqm(*xb*,*yb*,*Nx*,*Ny*,*vx*,*vy*)	=	test for equality of means
Eqv(*vx*,*vy*,*Nx*,*Ny*)	=	test for equality of variances
Eqp(*px*,*py*,*Nx*,*Ny*)	=	test for equality of proportions

where *xb* and *yb* are the means, *vx* and *vy* are the variances and *px* and *py* are the proportions.

Bowman-Shenton and Jarque-Bera Normality Test

The standard moment based test statistic against normality is

$$\chi^2[2] = \text{N}\ [(\text{m}_3/\text{s}^3)^2/6 + (\text{m}_4/\text{s}^4 - 3)^2/24].$$

To calculate this, use the function, Jbt(*variable*).

R17.9 Calculating Correlation Coefficients

There are several types of correlation coefficients that one might compute, beyond the familiar product moment measure. The nonparametric measures of rank correlation and of concordance are additional examples. One might also be interested in correlations of discrete variables, which are usually not measured by simple moment based correlations. The following summarizes the computations of several types of correlations with *LIMDEP*. Some of these are computed with **CALC**, as described earlier, while a few others are obtained by using certain model commands.

Pearson Product Moment Correlations for Continuous Variables

For any pair of variables,

Cor(*variable*,*variable*) = sample correlation.

Biserial Correlation Between Continuous and Binary Variables

For obtaining the correlation between a continuous variable, *x*, and a binary variable, *d*, one would use the 'biserial' correlation. It turns out that the biserial correlation is equal to the ordinary, Pearson product moment correlation. So no special function is created for this. Just use

```
CALC        ; List ; Cor(continuous variable x, binary variable d) $
```

to obtain a biserial correlation coefficient.

Tetrachoric Correlation Coefficients for Binary Variables

This is equivalent to the correlation coefficient in the following bivariate probit model:

$$y_1^* = \mu + \varepsilon_1, \qquad y_1 = 1(y_1^* > 0)$$

$$y_2^* = \mu + \varepsilon_2, \qquad y_2 = 1(y_2^* > 0)$$

$$(\varepsilon_1,\varepsilon_2) \sim N_2[(0,0),(1,1,\rho)]$$

The applicable literature contains a number of approaches to estimation of this correlation coefficient, some a bit ad hoc. We proceed directly to the implied maximum likelihood estimator. Fit this 'model' with

```
BIVARIATE PROBIT ; Quiet ; Lhs = y1,y2 ; Rh1 = one ; Rh2 = one $
CALC             ; List ; Rho $
```

The reported estimate of ρ is the desired estimate. *LIMDEP* notices if your model does not contain any covariates in the equation, and notes in the output that the estimator is a tetrachoric correlation. If the **; Quiet** is omitted from the bivariate probit command, the correlation coefficient will be reported in the estimation results. The results below show an example

```
-----------------------------------------------------------------------------
FIML Estimation of Tetrachoric Correlation
Dependent variable                  Y1Y2
Log likelihood function        -24.70694
Estimation based on N =     20, K =   3
Inf.Cr.AIC  =   55.414 AIC/N =    2.771
--------+--------------------------------------------------------------------
      Y1|                  Standard            Prob.      95% Confidence
      Y2|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index    equation for Y1
Constant|    -.67449**       .30469    -2.21  .0269    -1.27168   -.07730
        |Index    equation for Y2
Constant|    -.12566         .28106     -.45  .6548     -.67652    .42520
        |Tetrachoric Correlation between Y1       and Y2
RHO(1,2)|     .29207         .35914      .81  .4161     -.41183    .99598
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The preceding suggests an interpretation for the bivariate probit model; the correlation coefficient reported is the *conditional* (on the independent variables) tetrachoric correlation.

Tetrachoric Correlation Matrices

The computation in the preceding can be generalized to a set of M binary variables, $y_1,...,y_M$. The tetrachoric correlation matrix would be the $M \times M$ matrix, **R**, whose off diagonal elements are the ρ_{mn} coefficients described immediately above. There are several ways to do this computation, again, as suggested by a literature that contains numerous recipes. Once again, the maximum likelihood estimator turns out to be a useful device.

A direct approach would involve expanding the latent model to

$$y_1^* = \mu + \varepsilon_1, \quad y_1 = 1(y_1^* > 0)$$

$$y_2^* = \mu + \varepsilon_2, \quad y_2 = 1(y_2^* > 0)$$

$$...$$

$$y_M^* = \mu + \varepsilon_M, \quad y_M = 1(y_M^* > 0)$$

$$(\varepsilon_1,\varepsilon_2,...,\varepsilon_M) \sim N_M[\mathbf{0},\mathbf{R}]$$

The appropriate estimator would be *LIMDEP*'s multivariate probit estimator in which the Rhs for all M equations contained only a constant. **MPROBIT** can handle up to $M = 20$. The correlation matrix produced by this procedure is precisely the full information MLE of the tetrachoric correlation matrix. However, for any M larger than two, this requires use of the GHK simulator to maximize the simulated log likelihood, and is extremely slow. The received estimators of this model estimate the correlations pairwise, as shown above. For this purpose, the FIML estimator is unnecessary. The matrix can be obtained using bivariate probit estimates.

The following procedure would be useable:

```
NAMELIST    ; y = y1,y2,...,yM $
CALC        ; m = Col(y) $
MATRIX      ; r =  Iden(m) $
PROCEDURE $
DO FOR      ; 20 ; i = 2,m $
CALC        ; i1 = i - 1 $
DO FOR      ; 10 ; j = 1,i1 $
BIVARIATE   ; Quiet ; Lhs    = y:i, y:j ; Rh1 = one ; Rh2 = one $
MATRIX      ; r(i,j) = rho $
MATRIX      ; r(j,i) = rho $
ENDDO       ; 10 $
ENDDO       ; 20 $
ENDPROCEDURE $
EXECUTE $
```

A final note, the preceding approach is not fully efficient. Each bivariate probit estimates (μ_m,μ_n) which means that μ_m is estimated more than once when $m> 1$. A minimum distance estimator could be used to reconcile these after all the bivariate probit estimates are computed. But, since the means are nuisance parameters in this model, this seems unlikely to prove worth the effort.

Polychoric Correlation

The polychoric correlation coefficient is used to quantify the correlation between discrete variables that are qualitative measures. An appropriate description is that the discrete variables are discretized counterparts to underlying quantitative measures. We typically use ordered probit models to analyze such data. The polychoric correlation measures the correlation between $y_1 = 0,1,...,J_1$ and $y_2 = 0,1,...,J_2$. (Note, J_1 need not equal J_2.) One of the two variables may be binary as well. (If both variables are binary, we use the tetrachoric correlation coefficient described above.)

To compute the polychoric correlation for a pair of qualitative variables, we use *LIMDEP*'s bivariate ordered probit model. First, compute the starting values for the first ordered probit model.

```
ORDERED     ; Lhs = y1 ; Rhs = one $
MATRIX      ; b1 = b ; mu1 = mu $
```

Next, compute the starting values for second equation, ordered probit or binary probit. Use one of the following sets of commands.

```
ORDERED     ; Lhs = y2 ; Rhs = one $
MATRIX      ; b2 = b ; mu2 = mu $
```

or

```
PROBIT      ; Lhs = y2 ; Rhs = one $
MATRIX      ; b2 = b $
```

Then, fit the model.

```
ORDERED     ; Lhs = y1,y2 ; Rh1 = one ; Rh2 = one
            ; Start = b1,mu1,b2,mu2,0 $  Omit mu2 if y2 is binary
```

The set of starting values includes a zero for ρ. This may be omitted. Alternatively, if you have a specific value other than zero in hand, you may provide it.

For a simple example, we compute the polychoric correlation between self reported health status and sex in the health care usage data examined earlier. Results appear below.

```
ORDERED     ; Lhs  = hsat ; Rhs = one $
MATRIX      ; b1 = b ; mu1 = mu $
PROBIT      ; Lhs  = female ; Rhs = one $
MATRIX      ; b2 = b $
ORDERED     ; Lhs  = hsat,female
            ; Rh1 = one ; Rh2 = one ; Start = b1,mu1,b2,0 $
```

```
-----------------------------------------------------------------------------
Bivariate Ordered Probit Model
Dependent variable              BivOrdPr
Log likelihood function    -12561.56607
Restricted log likelihood  -12563.05400
Chi squared [  12 d.f.]         2.97586
Significance level               .99571
McFadden Pseudo R-squared      .0001184
Estimation based on N =   4481, K =  12
Inf.Cr.AIC  =25147.132 AIC/N =    5.612
-----------------------------------------------------------------------------
```

```
--------+--------------------------------------------------------------------
 NEWHSAT|                 Standard            Prob.      95% Confidence
  FEMALE|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Mean inverse probability for NEWHSAT
Constant|    2.17739***      .04831    45.07  .0000     2.08270   2.27208
        |Mean inverse probability for FEMALE
Constant|    -.03944**       .01875    -2.10  .0354     -.07620   -.00269
        |Threshold Parameters for Probability Model for NEWHSAT
  MU(01)|     .23979***      .03248     7.38  .0000      .17613    .30345
  MU(02)|     .54536***      .04154    13.13  .0000      .46395    .62677
  MU(03)|     .86716***      .04543    19.09  .0000      .77812    .95620
  MU(04)|    1.13200***      .04702    24.07  .0000     1.03984   1.22417
  MU(05)|    1.65205***      .04865    33.96  .0000     1.55671   1.74740
  MU(06)|    1.91515***      .04917    38.95  .0000     1.81879   2.01152
  MU(07)|    2.32040***      .04989    46.51  .0000     2.22262   2.41818
  MU(08)|    3.00506***      .05156    58.29  .0000     2.90401   3.10610
  MU(09)|    3.50351***      .05401    64.86  .0000     3.39765   3.60937
        |Polychoric Correlation for NEWHSAT  and FEMALE
RHO(1,2)|    -.03288*        .01907    -1.72  .0847     -.07026    .00450
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

+------------------------------------------------------------------+
|Cross Tabulation                                                  |
|Row variable is NEWHSAT  (Out of range 0-49:      0)              |
|Number of Rows = 11      (NEWHSAT  =  0 to 10)                    |
|Col variable is FEMALE   (Out of range 0-49:      0)              |
|Number of Cols =  2      (FEMALE   =  0 to  1)                    |
|Chi-squared independence tests:                                   |
|Chi-squared[  10] =   14.11593   Prob value =  .16777             |
|G-squared  [  10] =   14.12122   Prob value =  .16753             |
+------------------------------------------------------------------+
|                 FEMALE                                           |
+--------+--------------+------+                                   |
| NEWHSAT|      0      1| Total|                                   |
+--------+--------------+------+                                   |
|       0|     29     37|    66|                                   |
|       1|     26     26|    52|                                   |
|       2|     58     54|   112|                                   |
|       3|    107     89|   196|                                   |
|       4|    109    128|   237|                                   |
|       5|    333    347|   680|                                   |
|       6|    208    226|   434|                                   |
|       7|    388    330|   718|                                   |
|       8|    571    501|  1072|                                   |
|       9|    272    228|   500|                                   |
|      10|    210    204|   414|                                   |
+--------+--------------+------+                                   |
|   Total|   2311   2170|  4481|                                   |
+------------------------------------------------------------------+
```

Nonparametric Measures of Agreement of Rankings

These two statistics measure the amount of agreement among sets of ranks. The first of them is used to measure the correlation of a set of ranks.

Rkc(*variable*1,*variable*2) = the rank correlation of two variables

$$\rho = 1 - 6\,\Sigma_i\, d_i^2\,/n(n^2 - 1),\ d_i = variable1_i - variable2_i$$

An application that is becoming common in the literature is to measure the agreement of a set of efficiency rankings in a study of technical efficiency. The measures, themselves, often have no natural magnitude, but the comparison of firms to each other is useful. A common exercise is to compute the rankings of firms with two different measures, and search for high correlation of the two sets of ranks.

Cnc(*x1*,...,*xK*) = Kendall's coefficient of concordance of the *K* sets of rankings

$$W = 12\Sigma_i(S_i - \bar{S})^2/[nK^2(n^2 - 1)] \text{ where } S_i = \Sigma_k x_{k,i}$$

The coefficient of concordance is used to measure the degree of agreement among *n* individuals each of which has a set of *K* ranks. For example, consider a panel of *n* judges, each ranking a panel of *K* paintings or musicians. A large sample chi squared test of the null hypothesis that all *n* individuals are in 'concordance' may be based on

$$\chi^2[K(n\text{-}1)] \;=\; K(n\text{-}1)W \dashrightarrow \chi^2\,[\,K(n\text{-}1)]$$

R17.10 Augmented Dickey Fuller Test

The Adf function automates the Dickey Fuller test for unit roots in time series data. The syntax is

CALC ; Adf (variable, type, lags for augmentation) $

where *variable* is the single time series variable to be analyzed
type = 1, 2 or 3 for unit root, drift, trend, lags >= 0
lags for augmentation is the number of additional lagged values to include

Users are referred to any of the standard texts, e.g., Greene (2012, Chapter 21) for details. An example based on the investment series in the Grunfeld data follows:

CALC ; Adf (i,1,3) $

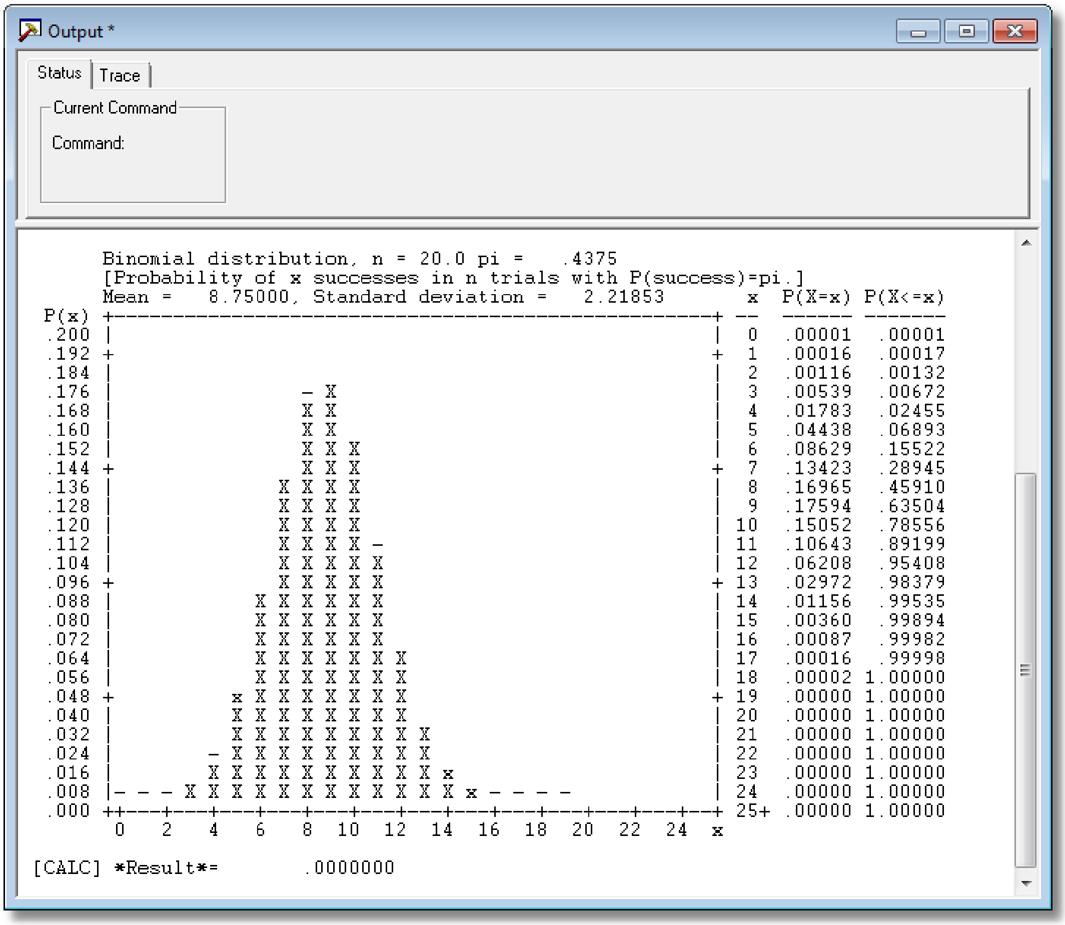
```
+-----------------------------------------------+
| Augmented Dickey Fuller Test for I            |
| Form: Random walk                             |
| Number of lagged differences in model is   3 |
| DF(tau) =   -2.32855, DF(gamma) =  -10.96038 |
| Critical values for   196 observations:       |
| DF(tau)                                       |
| 01 is  -2.58, .025 is  -2.23, .05 is  -1.95  |
| DF(gamma)                                     |
| 01 is -13.80, .025 is -10.50, .05 is  -8.10  |
+-----------------------------------------------+
```

R17.11 Plotting Discrete Distributions

There are a variety of tools that can be used to display probability distributions. Precise, accurate figures can be drawn by plotting the values of the probability density. Empirical approximations to probability distributions can be obtained by drawing histograms for large random samples of the random variable. **CALCULATE** provides numerous functions for computing continuous and discrete probabilities and densities from a variety of distributions. The following additional functions will produce tables and simple character based plots for discrete distributions:

Tbb(*p*,*n*)	for binomial probabilities with probability *p*, *n* trials,
Tbp(*lambda*)	for Poisson with mean *lambda*,
Tbg(*p*)	for geometric with parameter *p*,
Tbn(*p*,*n*)	for negative binomial with probability *p* and *n* successes,
Tbh(*p*,*m*,*n*)	for hypergeometric with probability *p*, population size *m*, *n* successes.

Calculating the function with specified parameters produces the listing and figure, as shown in the illustration below. **CALC ; List ; Tbb (.4375,20) $** produces the output in Figure R17.6.



```
     Binomial distribution, n = 20.0 pi =   .4375
     [Probability of x successes in n trials with P(success)=pi.]
     Mean =   8.75000, Standard deviation =   2.21853        x  P(X=x) P(X<=x)
P(x) +---------------------------------------------------+ --  ------ -------
.200 |                                                   |  0  .00001  .00001
.192 +                                                   +  1  .00016  .00017
.184 |                                                   |  2  .00116  .00132
.176 |                - X                                |  3  .00539  .00672
.168 |                X X                                |  4  .01783  .02455
.160 |                X X                                |  5  .04438  .06893
.152 |                X X X                              |  6  .08629  .15522
.144 +                X X X                              +  7  .13423  .28945
.136 |              X X X X                              |  8  .16965  .45910
.128 |              X X X X                              |  9  .17594  .63504
.120 |              X X X X                              | 10  .15052  .78556
.112 |              X X X X -                            | 11  .10643  .89199
.104 |              X X X X X                            | 12  .06208  .95408
.096 +              X X X X X                            + 13  .02972  .98379
.088 |            X X X X X X                            | 14  .01156  .99535
.080 |            X X X X X X                            | 15  .00360  .99894
.072 |            X X X X X X                            | 16  .00087  .99982
.064 |            X X X X X X X                          | 17  .00016  .99998
.056 |            X X X X X X X                          | 18  .00002 1.00000
.048 +          x X X X X X X X                          + 19  .00000 1.00000
.040 |          X X X X X X X X                          | 20  .00000 1.00000
.032 |          X X X X X X X X X                        | 21  .00000 1.00000
.024 |        - X X X X X X X X X                        | 22  .00000 1.00000
.016 |        X X X X X X X X X X x                      | 23  .00000 1.00000
.008 |- - - X X X X X X X X X X X X x - - - -            | 24  .00000 1.00000
.000 ++---+---+---+---+---+---+---+---+---+---+---+---+--+ 25+ .00000 1.00000
      0   2   4   6   8  10  12  14  16  18  20  22  24  x

[CALC] *Result*=       .0000000
```

Figure R17.6 Calculator Plot of Binomial Distribution

R18: Two Step Estimators

Fitting a two equation model in two steps is a common procedure. Heckman's (1979) sample selection model is a widely cited example. There are many other applications. The typical case involves computing a model at the first step, then inserting either a prediction or a residual (see Terza, Basu and Rathouz (2008)) in a second equation. Under the usual assumptions, the second step estimator is consistent, but the standard errors usually computed must be corrected for the inclusion of a variable that is based on the estimated parameters from the first step.

R18.1 Covariance Matrices for Two Step Estimation

The essential parts of the two step procedure are

Step 1. A model is estimated by least squares or maximum likelihood. Denote the parameters estimated at this step as $\boldsymbol{\theta}_1$.

Step 2. A second model is estimated in which a predicted value or a residual from the model in Step 1 appears on the right hand side of the equation. Denote the full set of parameters estimated at this step as $\boldsymbol{\theta}_2$.

We take it as given that estimation at both steps is consistent – the modeler will have to verify this on a case by case basis. The remaining computation then is the correction of the estimated asymptotic covariance matrix for the estimator at Step 2 for the randomness of the estimator from Step 1 which has been used in the computation. We base our results for this computation on the Murphy and Topel (1985) paper which presents a general method of doing the calculations. (See Greene (2012) for additional discussion.) (There are like results for GMM estimation – see Newey (1984) – however, we restrict our attention to maximum likelihood estimation in *LIMDEP*.)

The underlying result is as follows (again, from Greene): Let $\mathbf{V}_2$ be the uncorrected covariance matrix computed at Step 2, using the parameter estimates obtained at Step 1 as if they were known, and $\mathbf{V}_1$ be the estimator of the asymptotic covariance matrix for the parameter estimates obtained at Step 1. Both of these estimators are based on the respective log likelihood functions. In addition, define

$$\mathbf{C} = \sum_{i=1}^{n}\left(\frac{\partial \ln f_{i2}}{\partial \boldsymbol{\theta}_2}\right)\left(\frac{\partial \ln f_{i2}}{\partial \boldsymbol{\theta}_1\,'}\right)$$

and

$$\mathbf{R} = \sum_{i=1}^{n}\left(\frac{\partial \ln f_{i2}}{\partial \boldsymbol{\theta}_2}\right)\left(\frac{\partial \ln f_{i1}}{\partial \boldsymbol{\theta}_1\,'}\right)$$

(Note the derivatives shown are the derivatives of individual terms in the two log likelihoods.) With these in hand, the corrected covariance matrix for the second step estimator is

$$\mathbf{V}_2^* \;=\; \mathbf{V}_2 + \mathbf{V}_2[\mathbf{C}\mathbf{V}_1\mathbf{C}' - \mathbf{R}\mathbf{V}_1\mathbf{C}' - \mathbf{C}\mathbf{V}_1\mathbf{R}']\mathbf{V}_2$$

Since the variety of combinations of model specifications which can give rise to this computation is infinite, it is not possible to automate this generally in *LIMDEP*. But, a few special cases are automated. The following will list these cases, then suggest some approaches for doing the computation in other cases.

R18.2 Two Step Estimation for an Endogenous Discrete Variable

The general case that has been automated is a model of the form:

y_1 = a discrete variable specified by a probit, logit, Poisson, or negative binomial model or by a linear regression model,

y_2 = a dependent variable whose conditional mean function is a function of $E[y_1]$.

Models of this sort could in principle be estimated by full information maximum likelihood. We consider two step estimation instead, which is usually simpler. Models for which the second step shown above is automated are the following:

- probit and probit with heteroscedasticity,
- truncated regression,
- tobit and tobit with heteroscedasticity,
- Poisson and negative binomial regression,
- linear regression.

For these models, the estimation procedure is the following two steps:

```
PROBIT, LOGIT, etc. ; Lhs = y1 ; Rhs = as usual
                    ; Prob = py ←  ; Keep for Poisson or negative binomial
                    ; Hold  $
     Model name     ; Lhs = y2
                    ; Rhs = as usual,py ←  Note, py, not y1
                    ; 2Step = py $
```

In the example shown below, a probit model is estimated and the results are held for the second step. At the second step, linear and Poisson regression models are estimated. (Results for the probit model are omitted.) The second set of estimates in each example omit the Murphy and Topel correction. The correction seems to be inconsequential in the linear regression results. However, the same correction substantially changes the Poisson regression results.

```
     REGRESS        ; Lhs = docvis
                    ; Rhs = one,hhkids,hhninc,epublic
                    ; 2Step = epublic $
```

```
-----------------------------------------------------------------------------
Ordinary     least squares regression ............
LHS=DOCVIS   Mean                    =      2.87280
             Standard deviation      =      5.14529
             Number of observs.      =         4481
Model size   Parameters              =            4
             Degrees of freedom      =         4477
Residuals    Sum of squares          =      116455.
             Standard error of e     =      5.10019
Fit          R-squared               =       .01811
             Adjusted R-squared      =       .01745
Model test   F[  3,  4477] (prob)    =   27.5(.0000)
Diagnostic   Log likelihood          =  -13657.05554
             Restricted(b=0)         =  -13698.00656
             Chi-sq [  3]  (prob)    =  81.9(  .0000)
Info criter. Akaike Info. Criter.    =      3.25945
Covariance matrix corrected for two step using M&T ⟵
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
  DOCVIS|  Coefficient       Error        z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    1.20400*        .72845      1.65  .0984     -.22374   2.63175
  HHKIDS|    -.97872***      .15719     -6.23  .0000    -1.28680   -.67064
  HHNINC|   -1.12430**       .55245     -2.04  .0418    -2.20709   -.04152
 EPUBLIC|    2.78761***      .68587      4.06  .0000     1.44333   4.13190
--------+--------------------------------------------------------------------
Constant|    1.20400*        .72516      1.66  .0969     -.21729   2.62530
  HHKIDS|    -.97872***      .15710     -6.23  .0000    -1.28663   -.67081
  HHNINC|   -1.12430**       .54909     -2.05  .0406    -2.20051   -.04810
 EPUBLIC|    2.78761***      .68296      4.08  .0000     1.44904   4.12619
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
POISSON    ; Lhs = docvis
           ; Rhs = one,hhkids,hhninc,epublic
           ; 2Step = epublic $
```

```
-----------------------------------------------------------------------------
Poisson Regression
Dependent variable                 DOCVIS
Log likelihood function    -15988.45468
Restricted log likelihood  -16398.15386
Chi squared [   3 d.f.]       819.39836
Significance level               .00000
McFadden Pseudo R-squared      .0249845
Estimation based on N =   4481, K =   4
Inf.Cr.AIC  =31984.909 AIC/N =    7.138
Murphy/Topel 2Step VC matrix:P= EPUBLIC ⟵
Chi- squared = 38697.41953  RsqP= .0627
G  - squared = 23425.26493  RsqD= .0338
Overdispersion tests: g=mu(i)  : 10.260
Overdispersion tests: g=mu(i)^2: 10.487
```

```
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCVIS|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .18737         .65821     .28  .7759    -1.10269   1.47743
  HHKIDS|    -.35941***      .04667   -7.70  .0000     -.45087   -.26794
  HHNINC|    -.41892         .31052   -1.35  .1773    -1.02752    .18968
 EPUBLIC|    1.27807*        .65511    1.95  .0511     -.00592   2.56207
--------+--------------------------------------------------------------------
Constant|     .18737*        .10037    1.87  .0619     -.00936    .38410
  HHKIDS|    -.35941***      .01929  -18.63  .0000     -.39722   -.32160
  HHNINC|    -.41892***      .06919   -6.05  .0000     -.55453   -.28330
 EPUBLIC|    1.27807***      .09692   13.19  .0000     1.08811   1.46803
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

R18.3 Two Step Estimation for an Endogenous Regression Variable

The same set of procedures described in the previous section may also be used when the first step is a linear regression. Thus, the model is

$$y_1 \;=\; \theta_1'\mathrm{x}_1 + \boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_1 \sim \mathbf{N}[0,\sigma^2]$$

$$\text{density function for } y_2|\, y_1 = f\,(\,y_2, \mathbf{x}_2, |\, \boldsymbol{\theta}_2\,, \boldsymbol{\theta}_1'\mathbf{x}_1\,).$$

The model is estimated by fitting the first equation by linear least squares, then inserting the prediction from the linear model into the second equation and estimating $\boldsymbol{\theta}_2$ by maximum likelihood, including, presumably, the coefficient on $E[y_1]$ in the specification for y_2. The Murphy and Topel correction is then done at the second step. The only change will be the first step model command, which is changed from **PROBIT** or **LOGIT** to **REGRESS**. The command sequence would be

```
REGRESS     ; Lhs = y1 ; Rhs = as usual
            ; Keep = fy ← Note, ; Keep, not ; Prob
            ; Hold  $
Model       ; Lhs = y2
            ; Rhs = as usual,fy
            ; 2Step = fy $
```

The template shown immediately above is also the one used when the first step estimator is a Poisson or negative binomial regression model.

R18.4 Programming a Two Step Estimator

The preceding includes a fairly large number of possible specifications, given all of the different combinations. (Any of the first step models may be used with any of the second step models.) But, an essentially infinite number of possible different specifications remain. If you wish to use this procedure, you may have to program the second step correction yourself to do so. *LIMDEP's* various programming features should make this fairly easy. To illustrate, we will present two applications, first an extremely simple case of a probit first step and a linear second one, then a moderately complicated case in detail.

The first example has a probit first step equation:

$$y_1^* = \boldsymbol{\theta}'\mathbf{z} + u_1, , u_1 \sim N[0.1]$$

$$y_1 = \mathbf{1}(y_1^* > 0),$$

$$E[y_1] = \Phi(\boldsymbol{\theta}'\mathbf{z})\ \Phi(.) = \text{standard normal CDF}$$

and a second step linear regression model fit by least squares,

$$y_2 = \boldsymbol{\beta}'\mathbf{x} + \alpha E[y_1|\mathbf{z}] + \varepsilon.\ N[0,\sigma^2]$$

At step 1, $\boldsymbol{\theta}$ is estimated by maximizing the log likelihood

$$\log L_1 = \sum_{i=1}^{n} \log f_{1i}(y_{1i}, \mathbf{z}_i \mid \boldsymbol{\theta}) = \sum_{i=1}^{n} \log\Phi\left(q_i\boldsymbol{\theta}'\mathbf{z}_i\right), \text{ where } q_i = 2y_{1i} - 1.$$

At step 2, (β,α) are estimated by least squares regression of y_2 on x and $\Phi(\hat{\boldsymbol{\theta}}'\mathbf{z})$. The appropriate second step log likelihood for the regression is

$$\text{Log}L_2 = \sum_{i=1}^{n} -\tfrac{1}{2}\log\sigma^2 - \tfrac{1}{2}\log 2\pi - \tfrac{1}{2}(1/\sigma^2)[y_{i2} - \boldsymbol{\beta}'\mathbf{x}_i - \alpha\Phi(\hat{\boldsymbol{\theta}}'\mathbf{z}_i)]^2 .$$

The components of the covariance matrix are $\mathbf{V}_1$, the asymptotic covariance matrix estimated for the probit model at the first step, $\mathbf{V}_2$, the conventional covariance matrix at the second step – without the degrees of freedom correction, so $\mathbf{V}_2 = (\mathbf{e}'\mathbf{e}/n)(\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}$ where $\mathbf{X}^*$ contains the extra variable $\Phi(\hat{\boldsymbol{\theta}}'\mathbf{z}_i)$. From the definitions earlier,

$$\mathbf{C} = \sum_{i=1}^{n}\left(\frac{\partial \ln f_{i2}}{\partial\boldsymbol{\theta}_2}\right)\left(\frac{\partial \ln f_{i2}}{\partial\boldsymbol{\theta}_1{}'}\right) = \sum_{i=1}^{n}\left[\frac{e_i}{\sigma^2}\begin{pmatrix}\mathbf{x}_i\\ \Phi(\hat{\theta}'\mathbf{z}_i)\end{pmatrix}\right]\left[\frac{e_i}{\sigma^2}\alpha\phi(\hat{\theta}'\mathbf{z}_i)\mathbf{z}_i'\right]$$

and

$$\mathbf{R} = \sum_{i=1}^{n}\left(\frac{\partial \ln f_{i2}}{\partial\boldsymbol{\theta}_2}\right)\left(\frac{\partial \ln f_{i1}}{\partial\boldsymbol{\theta}_1{}'}\right) = \sum_{i=1}^{n}\left[\frac{e_i}{\sigma^2}\begin{pmatrix}\mathbf{x}_i\\ \Phi(\hat{\theta}'\mathbf{z}_i)\end{pmatrix}\right]\left[\frac{q_i\phi(q_i\hat{\theta}'\mathbf{z}_i)}{\Phi(q_i\hat{\theta}'\mathbf{z}_i)}\mathbf{z}_i'\right].$$

The following program computes this entirely from first principles, without using the preprogrammed estimation routines: The initial **NAMELIST** and **CREATE** commands set up the specific problem. The rest of the program is generic.

```
NAMELIST    ; z   = Rhs variables for the probit equation $
CREATE      ; y1 = Lhs variable for probit equation $
NAMELIST    ; x   = Rhs variables for the regression equation $
CREATE      ; y2 = Lhs variable for regression $

CALC        ; k1 = Col(z) $
CREATE      ; qi = 2 * y1 - 1 $
MAXIMIZE    ; Start= k1_0 ; Labels = k1_theta
            ; Fcn = Log(Phi(qi * theta1 ’ z)) $
MATRIX      ; v1 = varb ; theta = b $
CREATE      ; p1 = Phi(z ’ theta) $
NAMELIST    ; xs = x,p1 $
MATRIX      ; b_a = <xs ’ xs> * xs ’ y2 $
CREATE      ; ei = y2 - xs ’ b_a  $
CALC        ; s2 = ei ’ ei/n ; kplus1 = Col(xs) $
MATRIX      ; v2 = s2 * <xs ’ xs> $
CREATE      ; ci = 1/s2 * ei * 1/s2 * ei * b_a(kplus1) * N01(z ’ theta) $
CREATE      ; ri = 1/s2 * ei * qi * N01(qi * z ’ theta)/Phi(qi * z ’ theta) $
MATRIX      ; c = xs'[ci]z ; r = xs'[ri] z $
MATRIX      ; v2c = c*v1*c' - r*v1*c' - c*v1*r' ; v2c = v2 + v2*v2c*v2 $
MATRIX      ; Stat(b_a, v2c, xs) ; Stat(b_a, v2, xs) $
```

For the second example, we consider a multinomial logit model for a y_2 which has three outcomes and a y_1 determined by a probit model. The model is

$$y_1^* = \boldsymbol{\theta}'\mathbf{z} + \varepsilon_1,\ y_1 = \mathbf{1}(y_1^* > 0),\ E[y_1] = \Phi(\boldsymbol{\theta}'\mathbf{z}),\ \varepsilon \sim N[0,\sigma^2],\ \Phi(.) = \text{standard normal CDF},$$

$$\text{Prob}[y_2 = j] = e_j / (e_0 + e_1 + e_2),\ j = 0, 1, 2,$$

$$e_0 = 1$$

$$e_1 = \exp[\,\boldsymbol{\beta}_1'\mathbf{x} + \gamma_1\Phi(\boldsymbol{\theta}'\mathbf{z})\,]$$

$$e_2 = \exp[\,\boldsymbol{\beta}_2'\mathbf{x} + \gamma_2\Phi(\boldsymbol{\theta}'\mathbf{z})\,].$$

At step 1, $\boldsymbol{\theta}$ is estimated by maximizing the log likelihood

$$\begin{aligned}\log L_1 &= \textstyle\sum_{i=1}^{n} \log f_{1i}(y_{1i}, \mathbf{z}_i \mid \boldsymbol{\theta})\\ &= \textstyle\sum_{i=1}^{n} \log\Phi\left(q_i\boldsymbol{\theta}'\mathbf{z}_i\right),\ \text{where } q_i = 2y_{1i} - 1.\end{aligned}$$

After the first step is complete, the predictions, $\Phi(\boldsymbol{\theta}'\mathbf{z})$, are computed using the maximum likelihood estimates, then the log likelihood for the second model is maximized with respect to $\boldsymbol{\beta}_1,\gamma_1,\boldsymbol{\beta}_2,\gamma_2$ while treating the predictions as if they were observed data. The second step log likelihood function is

$$\begin{aligned}\log L_2 &= \sum\nolimits_{i=1}^{n} \log f_{2i}(y_{2i},\mathbf{x}_i,\Phi(\boldsymbol{\theta}'\mathbf{z}_i)\,|\,\boldsymbol{\beta}_1,\gamma_1,\boldsymbol{\beta}_2,\gamma_2)\\ &= \sum\nolimits_{i=1}^{n}\ \sum\nolimits_{j=0}^{2} d_{ij}\ \ln \text{Prob}[y_{2i}=j]\text{, where } d_{ij}=1 \text{ if } y_{2i}=j,\ j=0,1,2\end{aligned}$$

Each step produces its own estimated parameter vector and asymptotic covariance matrix. The matrices needed for the correction are:

$$\mathbf{C} = \sum\nolimits_{i=1}^{n} \begin{bmatrix} (d_{i1}-P_{i1})\begin{pmatrix}\mathbf{x}_i\\ \Phi(\boldsymbol{\theta}'\mathbf{z}_i)\end{pmatrix} \\ (d_{i2}-P_{i2})\begin{pmatrix}\mathbf{x}_i\\ \Phi(\boldsymbol{\theta}'\mathbf{z}_i)\end{pmatrix}\end{bmatrix} \times \left[(d_{i1}-P_{i1})\gamma_1+(d_{i2}-P_{i2})\gamma_2\right]\phi(\boldsymbol{\theta}'\mathbf{z}_i)\mathbf{z}_i'$$

$$\mathbf{R} = \sum\nolimits_{i=1}^{n} \begin{bmatrix} (d_{i1}-P_{i1})\begin{pmatrix}\mathbf{x}_i\\ \Phi(\boldsymbol{\theta}'\mathbf{z}_i)\end{pmatrix} \\ (d_{i2}-P_{i2})\begin{pmatrix}\mathbf{x}_i\\ \Phi(\boldsymbol{\theta}'\mathbf{z}_i)\end{pmatrix}\end{bmatrix} \times \left[\frac{q_i\phi(\boldsymbol{\theta}'\mathbf{z}_i)}{\Phi(q_i\boldsymbol{\theta}'\mathbf{z}_i)}\mathbf{z}_i'\right]$$

(Derivatives for the logit and probit log likelihoods above appear in greater detail later in this manual.)

The first part of the routine is set up for the particular application. The remainder is general and need not be changed.

```
NAMELIST    ; x   =… define the Rhs for the logit model
            ; z   =… define the Rhs for the probit model
CREATE      ; y1 =… dependent variable in probit model
            ; y2 =… dependent variable in logit model
```

Next, we estimate the probit model. The IMR = lambda is just for convenience. It computes the q*N01/Phi in the first log likelihood. We pick up the other terms now.

```
PROBIT      ; Lhs = y1 ; Rhs = z ; Prob = prob ; Hold(IMR = lambda) $
CREATE      ; den1 = N01(b'z) $
MATRIX      ; v1 = varb $
```

Augment the Rhs from the logit model with the fitted probability from the probit model, then fit the logit model.

```
NAMELIST    ; xp = x,prob $
LOGIT       ; Lhs = y2 ; Rhs = xp $
```

Get the subvectors of the logit parameter vector and the coefficients on the fitted probability.

```
CALC      ; k = Col(xp) ; j21 = k+1 ; j22 = 2*k
          ; gamma1 = b(k) ; gamma2 = b(j22) $
MATRIX    ; b1 = b(1:k) ; b2 = b(j21:j22) $
```

Compute the scalars that appear in the summations in the construction of the *c* and *r* matrices.

```
CREATE    ; d0  = (y2=0) ; d1 = (y2=1) ; d2 = (y2=2)
          ; e1  = Exp(b1'xp) ; e2 = Exp(b2'xp)
          ; p0  = 1 / (1 + e1 + e2 ) ; p1 = e1 * p0 ; p2 = e2 * p0
          ; u1  = (d1 - p1 ) ; u2 = (d2 - p2 )
          ; dc1 = u1*(u1*gamma1 + u2*gamma2)*den1 ; dr1 = u1*lambda
          ; dc2 = u2*(u1*gamma1 + u2*gamma2)*den1 ; dr2 = u2*lambda $
```

Note the matrix constructions. The namelist [variable] namelist format is specifically for computing matrices of the form of *c* and *r* in the expressions above. We compute both matrices in two parts, then stack the parts.

```
MATRIX    ; cm1 = xp' [dc1 ] z ; cm2 = xp' [dc2 ] z
          ; rm1 = xp' [dr1 ] z ; rm2 = xp' [dr2 ] z
          ; c = [cm1 / cm2 ] ; r = [rm1 / rm2 ]
```

The last computation computes the corrected covariance matrix, and then displays the results.

```
          ; t = c * v1 * c' – c * v1 * r' – r * v1 * c'
          ; v2 = varb + varb * t * varb
          ; Stat(b,v2) $
```

R18.5 Theory for Two Step Estimators

This section will present the theoretical results which underlie the Murphy and Topel (1985) estimator for the asymptotic covariance matrix of a two step maximum likelihood estimator. These results are used at many points in this manual.

For convenience, we will temporarily suppress explicit references to data and observations. We consider maximum likelihood estimation of two parameter vectors, $\boldsymbol{\theta}_1$ which appears in the first step log likelihood, log $L_1(\boldsymbol{\theta}_1)$, and $\boldsymbol{\theta}_2$ which appears with $\boldsymbol{\theta}_1$ in the second step log likelihood, log $L_2(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)$. The second step of the procedure is to maximize with respect to $\boldsymbol{\theta}_2$ the conditional log likelihood,

$$\log L_2^c = \log L_2\left(\hat{\boldsymbol{\theta}}_1, \boldsymbol{\theta}_2\right)$$

where $\hat{\boldsymbol{\theta}}_1$ is the first step maximum likelihood estimator computed by maximizing $\log L_1(\boldsymbol{\theta}_1)$. Maximizing log L_2^c with respect to $\boldsymbol{\theta}_2$ while inserting the previously obtained MLE, $\hat{\boldsymbol{\theta}}_1$ is not generally equivalent to FIML estimation of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ by maximizing log $L_2(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)$ with respect to $\boldsymbol{\theta}_1$, and $\boldsymbol{\theta}_2$ simultaneously.

The following useful result appears in Maddala (1983, p. 243)

$$\hat{\boldsymbol{\theta}}_2 - \boldsymbol{\theta}_2 \overset{a}{\sim} -\left(E\left[\frac{\partial^2 \log L_2}{\partial\boldsymbol{\theta}_2\partial\boldsymbol{\theta}_2'}\right]\right)^{-1}\left[\left(\frac{\partial \log L_2}{\partial\boldsymbol{\theta}_2}\right)+E\left[\frac{\partial^2 \log L_2}{\partial\boldsymbol{\theta}_2\partial\boldsymbol{\theta}_1'}\right]\left(\hat{\boldsymbol{\theta}}_1-\boldsymbol{\theta}_1\right)\right]$$

where the symbol $\overset{a}{\sim}$ means 'has the same asymptotic distribution as.' The matrix on the left would be the asymptotic covariance matrix of $\hat{\boldsymbol{\theta}}_2$ if $\boldsymbol{\theta}_1$ were not present in the log likelihood. In order to generate the population analog of the Murphy and Topel estimator, we now insert two other useful results from 'regular' maximum likelihood estimation. First,

$$E\left[\frac{\partial^2 \log L_2}{\partial\boldsymbol{\theta}_2\partial\boldsymbol{\theta}_1'}\right] = -E\left[\left(\frac{\partial \log L_2}{\partial\boldsymbol{\theta}_2}\right)\left(\frac{\partial \log L_2}{\partial\boldsymbol{\theta}_1'}\right)\right].$$

Second, if $\hat{\boldsymbol{\theta}}_1$ is the maximum likelihood estimator of $\boldsymbol{\theta}_1$ based on $\log L_1(\boldsymbol{\theta}_1)$, then

$$\hat{\boldsymbol{\theta}}_1 - \boldsymbol{\theta}_1 \overset{a}{\sim} -\left(E\left[\frac{\partial^2 \log L_1}{\partial\boldsymbol{\theta}_1\partial\boldsymbol{\theta}_1'}\right]\right)^{-1}\left(\frac{\partial \log L_1}{\partial\boldsymbol{\theta}_1}\right).$$

Insert these two expressions into Maddala's result to obtain

$$\hat{\boldsymbol{\theta}}_2 - \boldsymbol{\theta}_2 \overset{a}{\sim}$$

$$-\left(E\left[\frac{\partial^2 \log L_2}{\partial\boldsymbol{\theta}_2\partial\boldsymbol{\theta}_2'}\right]\right)^{-1}\left[\left(\frac{\partial \log L_2}{\partial\boldsymbol{\theta}_2}\right)-E\left[\left(\frac{\partial \log L_2}{\partial\boldsymbol{\theta}_2}\right)\left(\frac{\partial \log L_2}{\partial\boldsymbol{\theta}_1'}\right)\right]\left(-E\left[\frac{\partial^2 \log L_1}{\partial\boldsymbol{\theta}_1\partial\boldsymbol{\theta}_1'}\right]\right)^{-1}\left(\frac{\partial \log L_1}{\partial\boldsymbol{\theta}_1}\right)\right].$$

To find the asymptotic variance, note that this expression is of the form

$$\hat{\boldsymbol{\theta}}_2 - \boldsymbol{\theta}_2 \overset{a}{\sim} \mathbf{V}_2[\mathbf{g}_2 - \mathbf{A}\mathbf{V}_1\mathbf{g}_1]$$

where $\mathbf{g}_1$ and $\mathbf{g}_2$ are random (first derivative) vectors and $\mathbf{V}_2$, $\mathbf{V}_1$, and $\mathbf{A}$ are nonstochastic matrices (of expectations). Thus, the asymptotic variance would be

$$\text{Asy.Var}[\hat{\boldsymbol{\theta}}_2 - \boldsymbol{\theta}_2] =$$

$$\mathbf{V}_2\{\text{AVar}[\mathbf{g}_2] + \mathbf{A}\mathbf{V}_1\text{AVar}[\mathbf{g}_1]\mathbf{V}_1\mathbf{A}' - \mathbf{A}\mathbf{V}_1\text{ACov}[\mathbf{g}_1,\mathbf{g}_2] - \text{ACov}[\mathbf{g}_2,\mathbf{g}_1]\mathbf{V}_1\mathbf{A}'\}\mathbf{V}_2.$$

The two asymptotic variances in the expression are the negatives of the expected Hessians of the respective log likelihoods. $\mathbf{V}_2$ is what would normally be computed as the asymptotic covariance matrix for $\hat{\theta}_2$, but this expression corrects it for the presence of $\hat{\boldsymbol{\theta}}_1$. Thus, AVar[$\mathbf{g}_2$] = $\mathbf{V}_2^{-1}$ and Asy.Var[$\mathbf{g}_1$] = $\mathbf{V}_1^{-1}$. The asymptotic covariance of the two gradients is less familiar, but is estimable with the sample counterparts of the individual terms in the log likelihoods. The parts, therefore, can be seen to be the population analogs of the Murphy and Topel estimators presented earlier. We estimate $\mathbf{V}_2$ with the conventional (albeit incorrect) estimator of the asymptotic variance of $\hat{\theta}_2$. We estimate $\mathbf{V}_1$ with the estimator of the asymptotic covariance matrix of $\hat{\boldsymbol{\theta}}_1$. Finally, we have estimated $\mathbf{A}$ with $\mathbf{G}'\mathbf{M}$ and ACov[$\mathbf{g}_2$,$\mathbf{g}_1$] with $\mathbf{G}'\mathbf{D}$. (Factors 1/*n* have been omitted in several places.)

When the second step of the estimation is a least squares regression, we can obtain some simplification. The equation estimated by ordinary least squares is

$$y_{i2} = \gamma h(\mathbf{x}_{i1},\boldsymbol{\theta}_1) + \mathbf{x}_{i2}'\boldsymbol{\theta}_2 + \varepsilon_{i1}.$$

Define the $n\times(1+K_2)$ matrix $\mathbf{Z}_2 = [\mathbf{z}_i']_{i=1,...,n} = [h(\mathbf{x}_{i1},\boldsymbol{\theta}_1), \mathbf{x}_{i2}']_{i=1,...,n}$. Let $\mathbf{V}_1$ denote the $K_1\times K_1$ estimated asymptotic covariance matrix for the first step estimator of $\boldsymbol{\theta}_1$ and let $\mathbf{V}_2$ denote the estimated asymptotic covariance matrix for the second step least squares estimator of $[\gamma,\boldsymbol{\theta}_1]$. Typically, this would be $s^2\left[\hat{\mathbf{Z}}'\hat{\mathbf{Z}}\right]^{-1}$ where $s^2 = \mathbf{e}'\mathbf{e}/(n-K_1-1)$. The degrees of freedom correction is immaterial in what follows, as the results are asymptotic. If the second step is viewed as conditional maximum likelihood estimation, then s^2 would be replaced with $\hat{\sigma}^2 = \mathbf{e}'\mathbf{e}/n$. For purposes of the corrected covariance matrix, we would have

$$\mathbf{g}_{i2} = \frac{\varepsilon_i}{\sigma^2}\mathbf{z}_i,$$

$$\mathbf{m}_{i2} = \frac{\varepsilon_i}{\sigma^2}\gamma\frac{\partial h(x_{i1},\boldsymbol{\theta}_1)}{\partial\boldsymbol{\theta}_1} = \frac{\varepsilon_i}{\sigma^2}\gamma\mathbf{w}_i$$

$$\mathbf{d}_{i1} = \frac{\partial h(\mathbf{x}_{i1},\boldsymbol{\theta}_1)}{\partial\boldsymbol{\theta}_1}.$$

(The last is the *i*th term in the derivative of the log likelihood upon which the first step estimator of $\boldsymbol{\theta}_1$ is based.) In constructing the corrected covariance matrix,

$$\mathbf{G}'\mathbf{M} = \frac{\gamma}{\sigma^4}\sum_{i=1}^{n}\varepsilon_i^2\mathbf{z}_i\mathbf{w}_i'$$

Assuming that the disturbances are not heteroscedastic to begin with, the large sample behavior of this will be the same as that of $(\gamma/\sigma^2)\mathbf{Z}'\mathbf{W}$, so we will insert this in the corrected covariance matrix where appropriate. (A factor 1/*n* needed to make this converge has been suppressed; it will cancel out in the end.) The other matrix is

$$\mathbf{G}'\mathbf{D} = \frac{1}{\sigma^2}\sum_{i=1}^{n}\varepsilon_i\mathbf{z}_i\mathbf{d}_i'.$$

Combining terms, then, we use

$$\mathbf{V}_2^* = \sigma^2(\mathbf{Z}'\mathbf{Z})^{-1} + \sigma^2(\mathbf{Z}'\mathbf{Z})^{-1}[(\gamma/\sigma^2)^2(\mathbf{Z}'\mathbf{W})\mathbf{V}_1(\mathbf{W}'\mathbf{Z})$$

$$- \; (\gamma/\sigma^4)(\mathbf{Z}'\mathbf{W})\,\mathbf{V}_1\left(\sum_{i=1}^{n} \varepsilon_i \mathbf{d}_i \mathbf{z}_i'\right) - (\gamma/\sigma^4)\left(\sum_{i=1}^{n} \varepsilon_i \mathbf{z}_i \mathbf{d}_i'\right)\mathbf{V}_1(\mathbf{Z}'\mathbf{W})]\;\sigma^2(\mathbf{Z}'\mathbf{Z})^{-1}.$$

A factor $(1/\sigma^4)$ will cancel in the second part of the equation. If we write the summation as $\mathbf{D}'\mathbf{E}\mathbf{Z}$ where $\mathbf{E} = \text{diag}(\varepsilon_i)$, then the expression reduces to

$$\mathbf{V}_2^* = \sigma^2(\mathbf{Z}'\mathbf{Z})^{-1} + (\mathbf{Z}'\mathbf{Z})^{-1}[\gamma^2(\mathbf{Z}'\mathbf{W})\mathbf{V}_1(\mathbf{W}'\mathbf{Z}) -$$

$$\gamma(\mathbf{Z}'\mathbf{W})\mathbf{V}_1(\mathbf{D}'\mathbf{E}\mathbf{Z}) - \gamma(\mathbf{Z}'\mathbf{E}\mathbf{D})\mathbf{V}_1(\mathbf{Z}'\mathbf{W})](\mathbf{Z}'\mathbf{Z})^{-1}.$$

Finally, in many situations, ε_i will be uncorrelated with both $\mathbf{z}_i$ and $\mathbf{d}_i$ and with elements in $\mathbf{z}_i\mathbf{d}_i.'$.For example, $\mathbf{d}_i$ might be simple multiples of elements in $\mathbf{x}_{i2}$. In this case, the second and third terms in the brackets would become small in large samples, leaving for this special case,

$$\mathbf{V}_2^* = \sigma^2(\mathbf{Z}'\mathbf{Z})^{-1} + (\mathbf{Z}'\mathbf{Z})^{-1}[\gamma^2(\mathbf{Z}'\mathbf{W})\mathbf{V}_1(\mathbf{W}'\mathbf{Z})](\mathbf{Z}'\mathbf{Z})^{-1}$$

$$= \sigma^2(\mathbf{Z}'\mathbf{Z})^{-1}[(\mathbf{Z}'\mathbf{Z}) + (\gamma/\sigma)^2(\mathbf{Z}'\mathbf{W})\mathbf{V}_1(\mathbf{W}'\mathbf{Z})](\mathbf{Z}'\mathbf{Z})^{-1}.$$

R19: Programming with Procedures

R19.1 Introduction

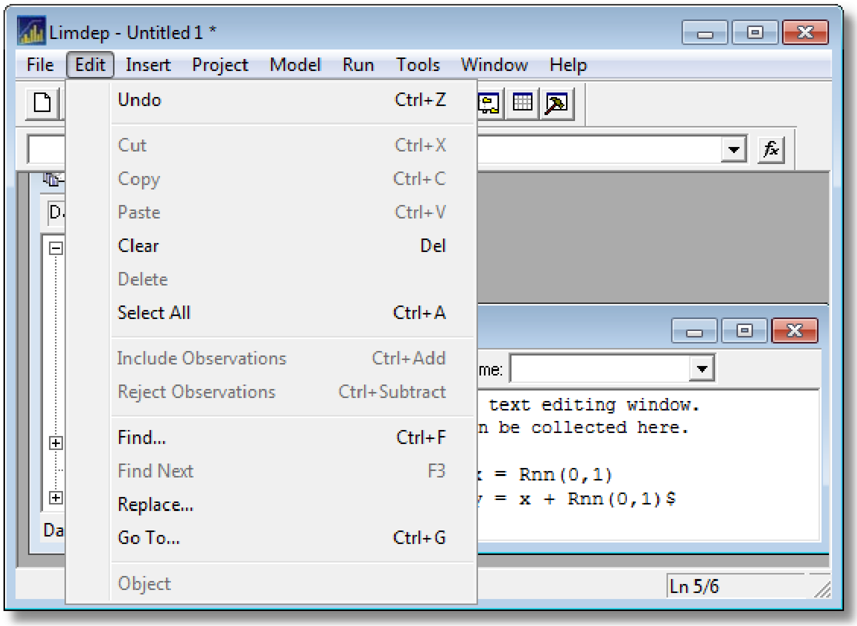
Your first uses of *LIMDEP* will undoubtedly consist of setting up your data and estimating the parameters of some of the models described in the *Econometric Modeling Guide*. The purpose of this chapter is to introduce *LIMDEP*'s tools for extending these estimators and writing new ones. The programs described in this and the next two chapters will also help you make more flexible use of the preprogrammed estimators, such as in testing hypotheses, analyzing specifications, and manipulating the results of the estimation procedures.

R19.2 The Text Editor

The tools and methods described in this chapter will make heavy use of the editing features of the program. The various menus described earlier and in the model sections to follow will be of limited usefulness when you are writing your own programs. The text editor will be essential.

R19.2.1 Placing Commands in the Editor

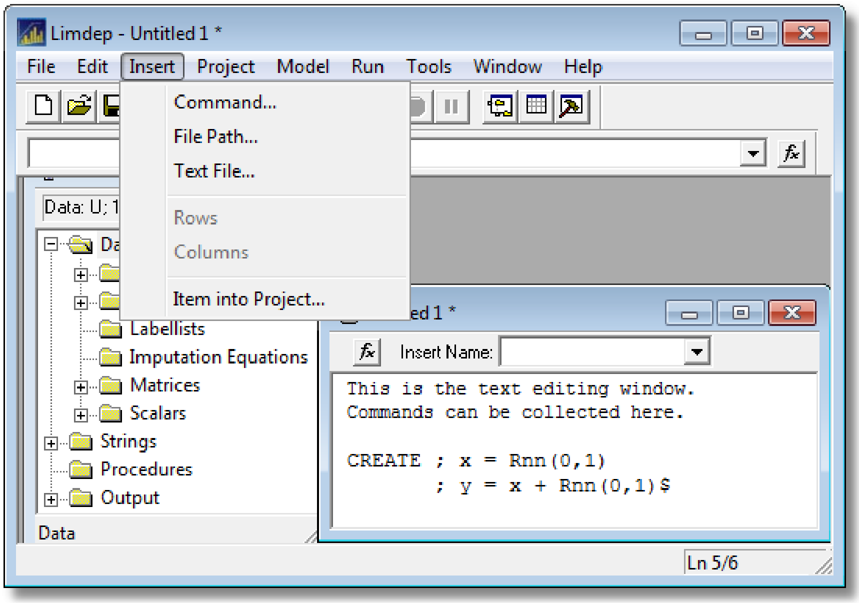
LIMDEP's editing window shown in Figure R19.1 is a standard Windows text editor. Enter text as you would in any other Windows based text editor. You may enter as much text as you like on the editing screen. The Edit menu provides standard editing features such as Cut, Copy, Paste, Go To, and Find.



```
 text editing window.
n be collected here.

: = Rnn(0,1)
/ = x + Rnn(0,1)$
```

Figure R19.1 The Editing Window and the Edit Menu

The Insert menu shown in Figure R19.2 can also be used in the editing window. The Insert menu allows you to place specific items on the screen in the editing window:



```
This is the text editing window.
Commands can be collected here.

CREATE  ; x = Rnn(0,1)
        ; y = x + Rnn(0,1)$
```

Figure R19.2 Insert Menu for Text Editor

- Insert:Command will place a specific *LIMDEP* command (verb) at the insertion point (where the cursor is). A dialog box allows you to select the verb from a full listing (with explanation) of the commands.

- Insert:File Path will place the full path to a specific file at the insertion point. Several *LIMDEP* commands use files. The dialog box will allow you to find the full path to a file on your disk drive, and insert that path in your command.

- Insert: Text File will place the full contents of any text file you select in the editor at the insertion point. You can merge command files, or create command files, using this tool.

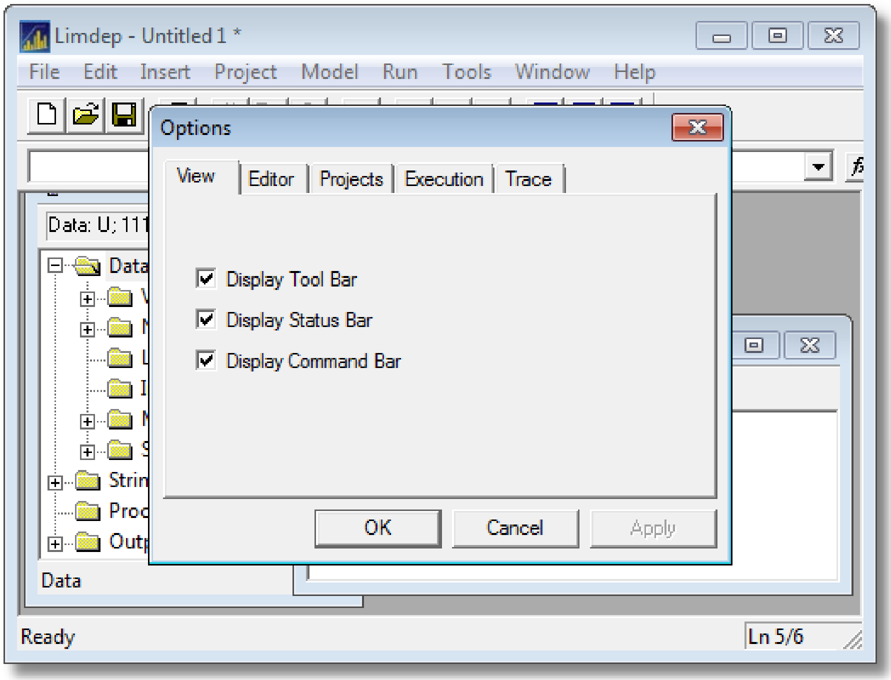
R19.2.2 Executing the Commands in the Editor

When you are ready to execute commands, highlight the ones you wish to submit. Then, you can execute the commands in one of two ways:

- Click the GO button on the *LIMDEP* toolbar. (If the toolbar is not displayed click the Tools:Options/View tab, then turn on the Display Tool Bar option. See Figure R19.3.)

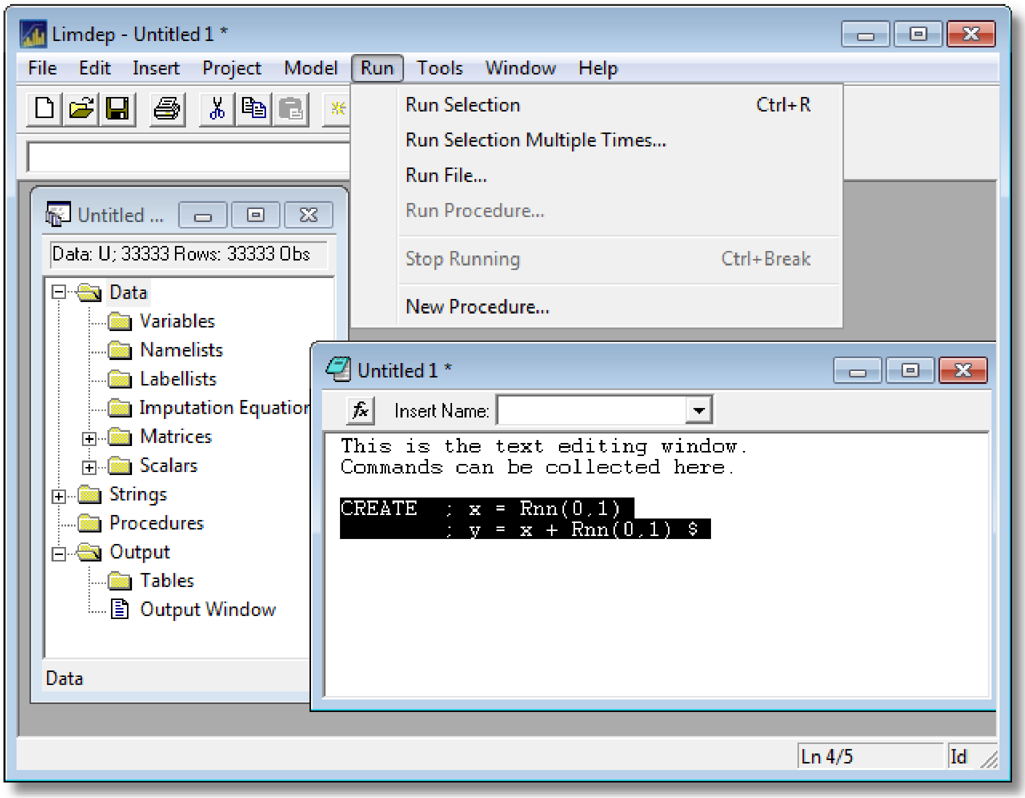
- Select the Run menu at the top of your screen. See Figure R19.4. When commands are highlighted, the first two items in this menu will be:
  - Run selection to execute the selected commands once.
  - Run selection Multiple Times to open a dialog box to specify the number of times to run the highlighted commands.

TIP: If you have not selected any lines in the editor, the two selections in the Run menu will be Run Line and Run Line Multiple Times. In this case, the line in question is the line where the cursor is located.



Display Tool Bar [✓]
Display Status Bar [✓]
Display Command Bar [✓]

Figure R19.3 Tools:Options/View Menu to Set Up Desktop



```
This is the text editing window.
Commands can be collected here.

CREATE   ; x = Rnn(0,1)
         ; y = x + Rnn(0,1) $
```

Figure R19.4 Run Menu

R19.2.3 Executing Model Commands Quietly

Quiet execution is useful when you are running an iterative program or executing a set of commands several times, and the output from any particular run is not of interest. You can turn off the output from model commands by placing the function **; Quiet** in your editor file at the point where you want the output turned off. For example, suppose you executed the second line in the following block of commands once, then the last three commands 100 times. First, initialize the procedure:

```
MATRIX      ; bsum = Init(2,1,0.0) $
```

Then, compute the average of 100 sets of least squares estimates.

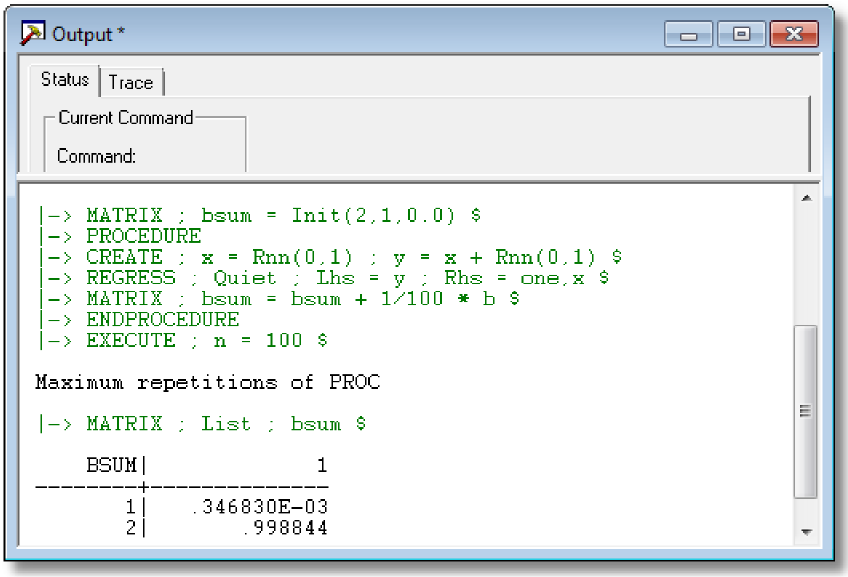
```
CREATE      ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
REGRESS     ; Lhs = y ; Rhs = x $
MATRIX      ; bsum = bsum + .01 * b $
```

This would produce a huge amount of regression output, though, in fact, only the average of the 100 estimators is really of interest. By adding the **; Quiet** function to the regression command, you can suppress the intermediate regression output. As shown, with **; Quiet**, this program would generate no visible output save for a small trace in the status window in the top half of your output window, which will show you what command is executing. The display in your output window will show you what command is executing at any point.

You might use this at the end of your procedure so that you can see the results that you have generated. For example, the preceding simulation, in full, might be:

```
MATRIX      ; bsum = Init(2,1,0.0) $
PROCEDURE $
CREATE      ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
REGRESS     ; Quiet ; Lhs = y ; Rhs = one,x $
MATRIX      ; bsum = bsum + 1/100 * b $
ENDPROCEDURE $
EXECUTE     ; n = 100 $
MATRIX      ; List ; bsum $
```

The output window shown in Figure R19.5 shows that even though 100 regressions were computed, the **; Quiet** function suppresses everything save for the last command which displays the only results of interest.



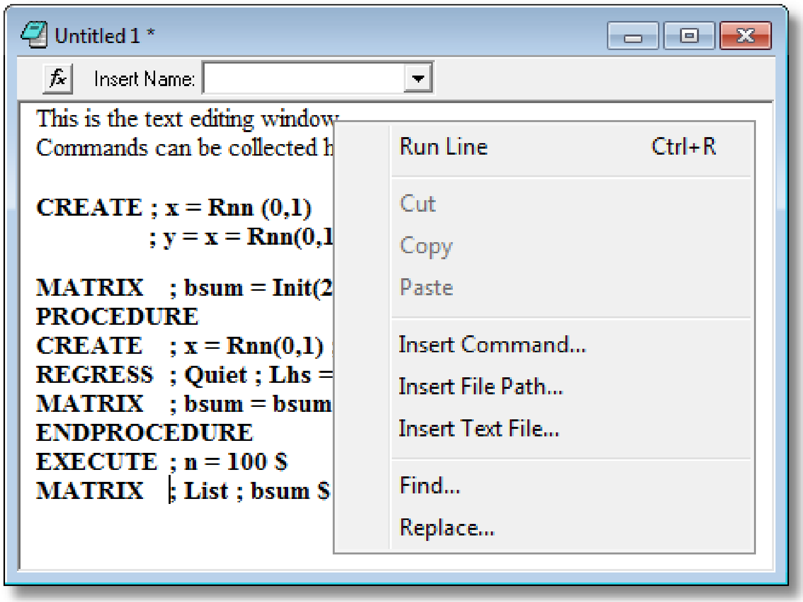
```
|-> MATRIX ; bsum = Init(2,1,0.0) $
|-> PROCEDURE
|-> CREATE ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
|-> REGRESS ; Quiet ; Lhs = y ; Rhs = one,x $
|-> MATRIX ; bsum = bsum + 1/100 * b $
|-> ENDPROCEDURE
|-> EXECUTE ; n = 100 $

Maximum repetitions of PROC

|-> MATRIX ; List ; bsum $

     BSUM|            1
---------+--------------
        1|   .346830E-03
        2|       .998844
```

Figure R19.5 Results from Quiet Procedure

R19.2.4 Using Text Files with the Editor

Reading a File into the Editor

You can insert the contents of any ASCII file into the editor. Just position the cursor in the editing window where you want the file inserted, then select Insert:Text File from the Insert menu and double click the file name. You can also insert a file if you right click in the editing window. This opens a menu that combines parts of the Edit, Insert and Run menus. From this menu, select Insert Text File to place the contents of the file in your editing window. See Figure R19.6.



```
This is the text editing window
Commands can be collected h

CREATE ; x = Rnn (0,1)
          ; y = x = Rnn(0,1

MATRIX    ; bsum = Init(2
PROCEDURE
CREATE    ; x = Rnn(0,1)
REGRESS   ; Quiet ; Lhs =
MATRIX    ; bsum = bsum
ENDPROCEDURE
EXECUTE  ; n = 100 $
MATRIX    ; List ; bsum $
```

Figure R19.6 ‘Right Click’ Menu in Editing Window

You can use edit/copy and edit/paste to move anything from your word processor into the text editor in *LIMDEP*, including equations, figures, etc. For example, if you paste *MathType* equations from a *Word* document, they will be replicated in the editing window. However, nonASCII items, such as these equations, will disappear when the contents of the text editor are saved on your computer at the end of your session.

Writing a File from the Editor

You can save the contents of the editor in an ASCII file as well. When the editing window is open, you can select Save or Save As from the File menu. The next dialog box will query you where to save the file. Also, when you leave the program, you will be queried if you wish to save the contents of the editor – this is usually called 'Untitled 1' unless you have opened an existing file. (And, note once again, if the window contains nonASCII items, these will be lost when the file is written in text format.)

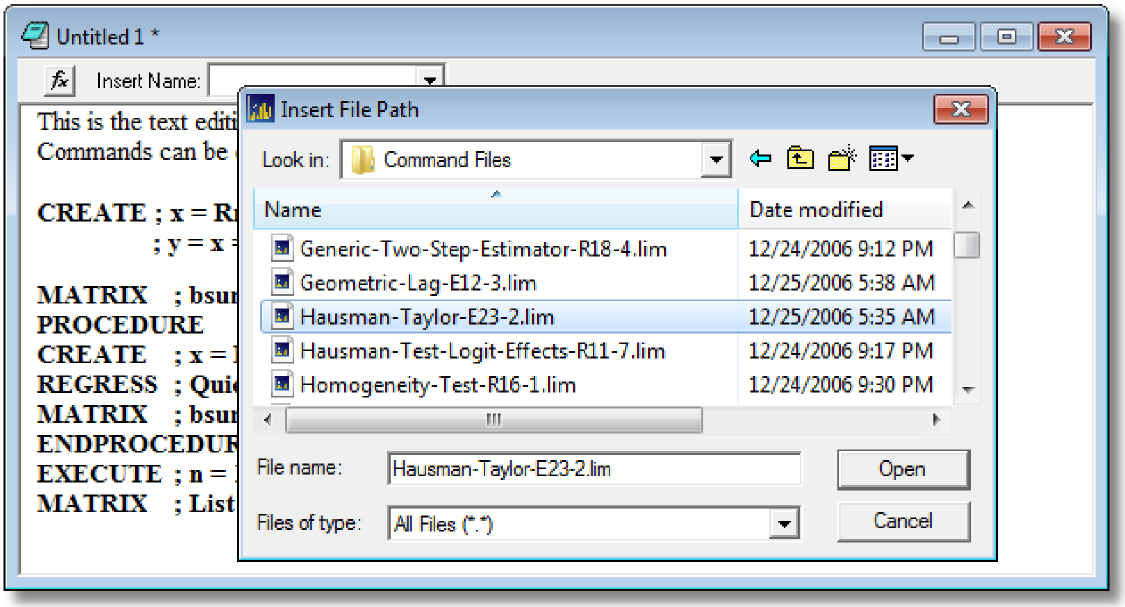
Executing a File with Run

To execute the commands in a file, select Run File from the Run menu. (See Figure R19.4) Double clicking the file name will automatically execute the commands in the file. This mode is similar to 'batch mode' in that commands are read from the file and executed as they are read, but they are not read into the text editor at the same time, and there is no interaction between you and the program while this is being done. Once the file is read and the commands are executed, the results appear in the output window and focus returns to the text editor as *LIMDEP* awaits your next instruction. The command files that you submit in this fashion may contain any commands or sets of commands that might otherwise be placed in the editor.

The command

OPEN ; input = the name of the file $

is the same as selection of the file from the Run menu. Insert:File Path will be useful with this command. The dialog box allows you to find the full path on your disk drive and insert that path in your command. Also, file names should be enclosed in double quotes. Insert:File Path does this automatically. See Figure R19.7 for an application of this device.



This is the text editi
Commands can be

```
CREATE ; x = R
       ; y = x
MATRIX   ; bsu
PROCEDURE
CREATE   ; x =
REGRESS  ; Qui
MATRIX   ; bsu
ENDPROCEDUR
EXECUTE  ; n =
MATRIX   ; List
```

Name	Date modified
Generic-Two-Step-Estimator-R18-4.lim	12/24/2006 9:12 PM
Geometric-Lag-E12-3.lim	12/25/2006 5:38 AM
Hausman-Taylor-E23-2.lim	12/25/2006 5:35 AM
Hausman-Test-Logit-Effects-R11-7.lim	12/24/2006 9:17 PM
Homogeneity-Test-R16-1.lim	12/24/2006 9:30 PM

File name: Hausman-Taylor-E23-2.lim

Files of type: All Files (*.*)

Figure R19.7 Using Insert File Path in the Text Editor

You may place a sequence of **OPEN** commands in your editing window if you wish. The text editor can open as many files as desired, but only one at a time. (Command files may not open other command files.)

R19.3 Estimation Programs and Postprocessing

For basic estimation purposes, the data setup and model commands,

READ	to input the data,
CREATE	to transform the data,
SAMPLE, REJECT, INCLUDE, PERIOD	to define the current sample, and
REGRESS, PROBIT, LOGIT ...	to estimate the model,

produce listings of parameter estimates, standard errors, and numerous diagnostic statistics. But, every estimation program also produces an easily 'retrievable' set of statistics and results for you to use in 'postprocessing.' By 'retrievable,' we mean that the number or set of numbers is available to you to use, symbolically, in a subsequent command. (This might seem routine, but, in fact, it is not at all. In software which is strictly menu driven, there is usually no means by which earlier results can be recovered for any purpose. This is a critical disadvantage of strictly menu driven statistical packages.) For example, consider testing the hypothesis that a set of coefficients equals zero in a probit model, using the likelihood ratio test. One could proceed as follows:

Step 1. Estimate the unrestricted model and write down the log likelihood, $\log L_u$.
Step 2. Estimate the restricted model and write down the log likelihood, $\log L_r$.
Step 3. Using a hand calculator, compute $\chi^2 = -2(\log L_r - \log L_u)$.

If the log likelihood functions were retrievable within the program, there would be no need to employ outside resources such as a calculator or a pencil and paper. Retrievability brings a second, less obvious benefit. There must be some means of using the result once it is retrieved. For the example above, for the log likelihoods to be useful in further computations, it must be possible to do the computation in Step 3 within the program. A close look at most modern econometrics packages reveals that they generally provide some means of manipulating scalar results such as log likelihood functions, once they are computed. This is the crucial function of 'programmability.' We have used this feature repeatedly in almost every chapter of this manual.

Programs do differ in the degree to which one can program and postprocess results. Consider, for example, carrying out a Hausman test on a subset of the coefficients estimated by two different estimators. This operation would require:

1. The ability to save, then retrieve both coefficient vectors and covariance matrices,
2. The ability to extract subvectors and submatrices in a way that leaves them accessible later,
3. The ability to manipulate simultaneously several matrices and vectors, and,
4. A means of evaluating the significance of the test statistic.

These require a considerable amount of flexibility. *LIMDEP*'s programming tools, notably the **CALC** and **MATRIX** commands, are written to provide the maximum access to estimation results and the greatest ability to manipulate those results.

All the estimation programs in *LIMDEP* produce four types of retrievable results:

1. Column vectors of data, in the form of predictions, residuals, or other functions,
2. Scalar results, such as log likelihood functions, test statistics, and parameters,
3. Matrix results, including coefficient vectors and asymptotic covariance matrices,
4. Coefficient estimates and a set of symbolic labels that facilitate testing hypotheses.

The features described in the next three sections rely heavily on this aspect of *LIMDEP*. What is referred to above as 'postprocessing' is the use of these estimation results in subsequent commands to analyze data in the context of a particular model, an extension of the model, or some general modeling framework. For example, *LIMDEP* provides the full set of tools needed to carry out the Hausman test suggested above, say, in the context of the discrete choice or nested logit model.

The primary commands for numeric manipulation of data and estimation results are:

1. **CREATE** which transforms a full column of data at a time,
2. **CALC** which computes single results, in standalone expressions or using earlier calculations, prior estimation results, columns of data, and so on, and
3. **MATRIX** for manipulating estimation results or other matrix valued results.

CREATE is described in Chapters R4 and R5. **MATRIX** and **CALC** are detailed in Chapters R16 and R17. The three commands will often be used at the same time. For example, as part of the analysis of the sample selection model described in Chapter E52, we require a statistic

$$\bar{\delta} = (1/n)\Sigma_i\lambda_i(z_i + \lambda_i)$$

where $z_i = \boldsymbol{\alpha}'\mathbf{w}_\mathrm{i}$ for a parameter vector $\boldsymbol{\alpha}$ and observation vector $\mathbf{w}_\mathrm{i}$, and

$\lambda_i = \phi(z_i)/\Phi(z_i)$, where ϕ and Φ are the standard normal PDF and CDF.

The coefficients are the estimation results from a **PROBIT** command. The sequence of commands which involves estimation and postprocessing, are

```
NAMELIST    ; wi = the list of names... $
PROBIT      ; ... ; Rhs = wi  $
MATRIX      ; alpha = b $      retrieve the coefficients
CREATE      ; zi = z’alpha ? compute the variable
            ; lambdai = N01(zi)/Phi(zi)
            ; deltai = lambdai * (zi + lambdai) $
CALC        ; deltabar = Xbr(deltai) $
```

(There are somewhat shorter ways to obtain this result.) This simple example suggests what most of your postprocessing programs will look like. To continue the example, having computed *deltabar*, you might use it (as we do) in a matrix result such as

$$\mathbf{V} = (s^2 + \beta_k^{\,2}\bar{\delta})[\mathbf{X}'\mathbf{X}]^{-1}.$$

The commands could be

```
NAMELIST    ; x = ... $
CALC        ; k = Col(x) $    number of columns
REGRESS     ; ... $           computes retrievable coefficient vector, B.
MATRIX      ; v = {ssqrd + b(k)^2 * deltabar} * <X’X> $
```

and, so on. Note, once again, how results are carried ‘downstream’ into subsequent commands.

R19.4 Procedures

LIMDEP operates primarily as an ‘interpreter.’ This means that commands are submitted one at a time, and carried out as they are received. This is as opposed to a ‘compiler’ which would assemble a number of commands in some fashion, translate them into its own language, then execute them all at once. ‘Batch’ mode, or batching commands provides a middle ground between these, whereby you can submit groups of commands from input files or as streams of commands from the editor or in a procedure. If the use of this is merely to submit a sequence of commands with a small number of keystrokes (for which *LIMDEP* provides several methods), then batching provides nothing more than a convenience. But, *LIMDEP* also provides batch like capabilities which make it operate more like a compiler than an interpreter. Consider the logic of an iterative program:

Step 1. Initial setup.
Step 2. Compute a result based on current information and previous results.
Step 3. Decide whether to exit the iteration or return to Step 2, and act on the decision.

In order to carry out such a sequence of commands, you must have several capabilities available. First, results of Steps 1 and 2 must be retrievable. Second, it must be possible not only to submit the set of commands in Step 2 in a batch mode, it must be possible to do so repeatedly. Step 3 may call for many repetitions of the same set of commands. Here is a trivial example:

```
Step 1. CALC        ;  i = 0 $
Step 2. CALC        ;  List ;  i = i + 1 $
Step 3. If i ≤  10, go to Step 2.
```

If we execute this program, it will display the numbers 1 to 10. The problem of retrievability is obviously solved, assuming, of course, that **CALC** can compute and define something called '*i*' in such a way that later on, *i* will exist. (Certainly it can; see the previous chapter.) The second step will be carried out 10 times. Obviously, you could simply *be* the program. That is, type the command and look at *i*. If *i* is less than or equal to 10, type it again. The point of this discussion is to devise a way to make *LIMDEP* do the repetitions for you.

As noted, *LIMDEP* provides several methods of batching commands. The example above could be handled as follows:

```
CALC          ; i = 0 $
PROCEDURE $
CALC          ; i = i + 1 $
ENDPROCEDURE $
EXECUTE       ; n = 10 $
```

This example initializes *i*, stores the updating command, then executes the stored command 10 times. There are other ways to do this, as well. For example, a shorter way to display the numbers from 1 to 10 is

```
PROCEDURE $
CALC          ; List ; i $
ENDPROCEDURE $
EXEC          ; i = 1,10 $
```

Yet another way to proceed would program the steps literally. This would be

```
CALC          ; i = 1 $
PROCEDURE $
LABEL         ; 100 $
CALC          ; List ; i ; i =  i + 1 $
GO TO         ; 100 ; i <= 10 $
ENDPROCEDURE $
EXECUTE
```

This procedure is only executed once, but it contains a *loop* within it. It displays, then updates *i* 10 times.

The device used in each case (and generally) will be the 'procedure.' Procedures such as these provide a convenient way to store commands. The **EXECUTE** command offers numerous options for how to carry out the procedure and how to decide to exit from the procedure.

LIMDEP is highly programmable. As shown in numerous examples already, and throughout the *Econometric Modeling Guide*, you can arrange long sequences of commands to perform intricate analyses. Procedures, which are similar to 'subroutines' or small programs, greatly extend this capability. Procedures will allow you to automate new estimators that are not already present in *LIMDEP*, and to compute certain test statistics that are not routine parts of the standard output. The remainder of this chapter will show you how to write and execute procedures.

R19.5 Defining and Executing Procedures

To store a set of commands you begin with the command

PROCEDURE or just **PROC**

This tells *LIMDEP* that the commands that will follow are not to be executed at the time, but just stored for later use. The end of a procedure is indicated with

ENDPROCEDURE or just **ENDPROC**

Once a set of commands has been entered as a procedure, you can execute it with

EXECUTE or just **EXEC**

The **EXECUTE** command has a number of options which are discussed below.

A procedure can be entered at any point, just by submitting it from the editing window. For example

```
CREATE      ; x = Rnn(0,1) ; y = x + 1 + Rnn(0,2) $
PROC $
SAMPLE      ; first - last $
REGRESS     ; Lhs = y ; Rhs = one,x $
ENDPROC $
CALC        ; first = 1 ; last = 10 $
EXEC
```

At the time the procedure is created, the sample limits might not exist. The procedure is defined, the sample limits are set, and, finally, the procedure is executed. The procedure, in turn, sets the sample and computes a regression.

You can also load a procedure from an input file. The file must contain the command **PROCEDURE** at the point at which the procedure is to begin, and **ENDPROCEDURE** at the end of the procedure. These might be the first and last commands in the file if you want only to input a procedure. If you **OPEN** such an input file, it will simply be loaded into the procedure buffer, exactly as if you had typed it. But, remember, the **PROCEDURE** cannot **OPEN** any files itself.

The following apply to procedures:

- The procedure loader is not a compiler. The commands you type are not checked in any way for validity. If you type nonsense, *LIMDEP* will dutifully store it for you. The problems will show up when you try to execute the procedure. (But, see below, procedures can be edited.)

- A procedure may consist of no more than 10,000 nonblank characters. When the commands are stored, the embedded blanks are removed and comments are stripped off. Still, it may pay to use short names and always use the four letter convention for model commands.

- The procedure may contain up to 50 commands, but remember that you can combine many **CREATE**, **CALC**, or **MATRIX** operations in a single command by separating them with semicolons.

- Only one active procedure can be defined at a time. If you issue a **PROCEDURE** command, any procedure which existed before is immediately erased. But, you can store up to 10 more procedures in a library, which is described in the next section.

- Project files (.lpj files) always contain not only the active procedure, but also any procedures that you have stored in your procedure library. They become part of the project.

R19.5.1 The Procedure Library

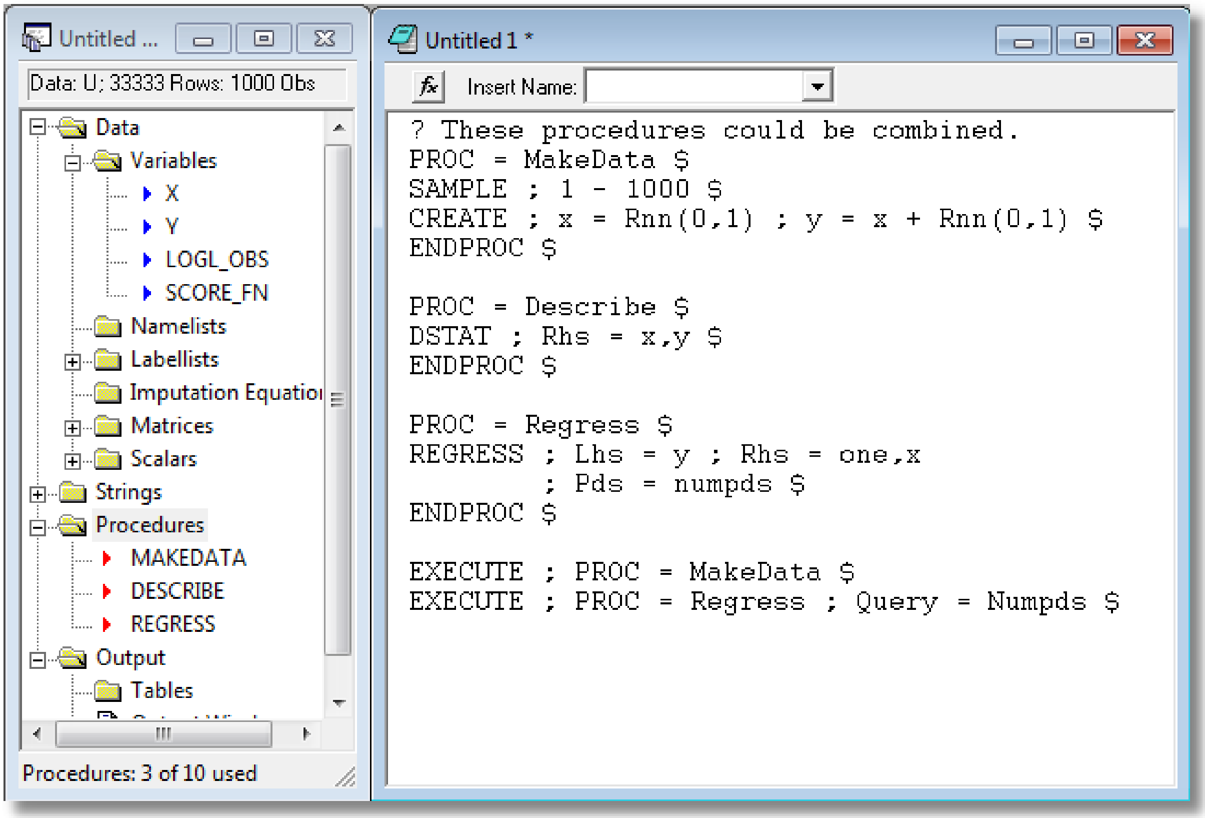
The preceding shows how to store an 'active' procedure that you can execute with the simple command **EXECUTE**. We denote this the 'current procedure.' You can also store up to 10 additional procedures, by name, in a library of procedures. Use

```
PROC =  procname $
```

to begin storage of a named procedure. The named procedure is then executed with

```
EXECUTE      ; Proc = procname ; ... $
```

(There are other options available for the **EXECUTE** command.) Therefore, you can have up to 11 procedures active at any time, the current procedure and up to 10 library routines. The current procedure can be invoked with the **EXECUTE** command, while the library procedures are executed by name. You will find the names of library procedures in your project window, as shown in the example below.



```
? These procedures could be combined.
PROC = MakeData $
SAMPLE ; 1 - 1000 $
CREATE ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
ENDPROC $

PROC = Describe $
DSTAT ; Rhs = x,y $
ENDPROC $

PROC = Regress $
REGRESS ; Lhs = y ; Rhs = one,x
        ; Pds = numpds $
ENDPROC $

EXECUTE ; PROC = MakeData $
EXECUTE ; PROC = Regress ; Query = Numpds $
```

Figure R19.8 Project Window with Procedure Library

R19.5.2 Executing a Procedure

The simplest means of executing a procedure you have stored is the command

EXECUTE

This will carry out the set of commands you have stored exactly once. At the end of the last command, the message 'Maximum repetitions' will appear on your screen. (You have requested one repetition.)

You can **EXECUTE** a procedure as many times as you like, just by repeating this command. A library procedure is executed with

EXECUTE ; Proc = procname $

NOTE: When you execute a library procedure, it becomes the current procedure. Thus, if you want to execute it again, the '**; Proc = procname**' is not necessary.

The other options for executing procedures in different ways are described in the following sections.

Controlling the Current Sample

Procedures will usually include commands that manipulate the active data set, such as model commands. You may change the current sample as part of your procedure as well. At the time the **EXECUTE** command is run, the sample in place will be the current sample however set. You can change this during the procedure execution slightly the way that other model commands do. The general syntax is

EXECUTE ; For [variable = value] ; … $

For example, if your data contain a binary variable, *union*, the following defines the sample to be operated on to be those observations with *union* equal to one, regardless of what the sample was before:

EXECUTE ; For [union = 1] ; … $

You can also control the sample with the following settings:

EXECUTE ; Sample = All ; … $
EXECUTE ; Sample = current ; … $ (which should not change anything)
EXECUTE ; Sample = the name of a binary variable, so that the sample is set to the observations for which the variable equals one.
EXECUTE ; Sample = i1, i2 ; … $ which is equivalent to **SAMPLE ; i1 – i2 $**

Ignore Warnings During Execution

Typically, when you execute a procedure, the program will stop the execution if an error occurs. For example, your procedure might contain a **REGRESS** command to compute a regression. If the variables in the regression are perfectly collinear, the model will be inestimable and an error will occur. At that point, your **EXECUTE** command will generally be aborted. You can direct the program to ignore warnings and just continue running with

; IgnoreWarnings

in the **EXECUTE** command.

Debugging Your Procedure

If your procedure is involved or lengthy it may be helpful to try executing it one line at a time, at least while you are developing it. That way, you can examine the execution one line at a time to find any errors in the commands. If you include

; Debug

in your **EXECUTE** command, the iteration will stop after each command is executed, and query you with a dialog box to see if you would like to continue or exit the procedure.

R19.5.3 Repeated Execution of a Procedure

There are several options available for the **EXECUTE** command that are based on use of the **PROCEDURE** as an iterative routine, such as that shown in the examples below. Repeated execution is obtained with

```
EXECUTE     ; n = number of repetitions $
```

The procedure will be executed the specified number of times unless some other method of exiting it is invoked. For example, consider the following:

```
CALC        ; count = 0 $
PROC $
CALC        ; count = count + 1 $
ENDPROC $
EXECUTE     ; n = 10 $
```

This will display the numbers from one to 10, then exit on maximum repetitions of 10.

R19.5.4 Execution with a Scalar Parameter

You can execute a procedure while carrying a single value of a parameter into the procedure. This would be useful for exploratory work. The command structure is

```
EXECUTE     ; name = value $
```

'*Name*' is created as a scalar which can be used (but not changed) by **CALC** while the procedure is executing. For example, the following computes Box-Cox regressions for different values of the transformation parameter:

```
PROC $
BOXCOX      ; Lhs = y ; Rhs = ... ; lambda = ll $
CALC        ; List ; ll $
ENDPROC $
EXECUTE     ; ll = .35 $
EXECUTE     ; ll = 1.294 $
```

and so on. This is a device that allows you to experiment with model specifications. For example, the following varies the correlation parameter in a bivariate probit model:

```
NAMELIST    ; x1 = ... first Rhs for bivariate probit model
            ; x2 = ... second Rhs for bivariate probit model $
CALC        ; k1 = Col(x1) ; k2 = Col(x2) ; rho12 = 0 $
PROC $
BIVARIATE PROBIT ; ... variables setup
            ; Rst = k1_b , k2_c , rho12 $
ENDPROC $
EXECUTE     ; rho12 = value $
```

R19.5.5 Query for a Parameter to Use in the Procedure

Your procedure may use a scalar value which you would like to vary according to the output you see on your screen. Or, you may wish to experiment with different oddly spaced values. For example, you might wish just to execute the procedure with different values of the scalar. The ridge regression estimator given in an example below depends on a scalar '*r*' which we supply to the routine each time it is carried out. This option operates as follows:

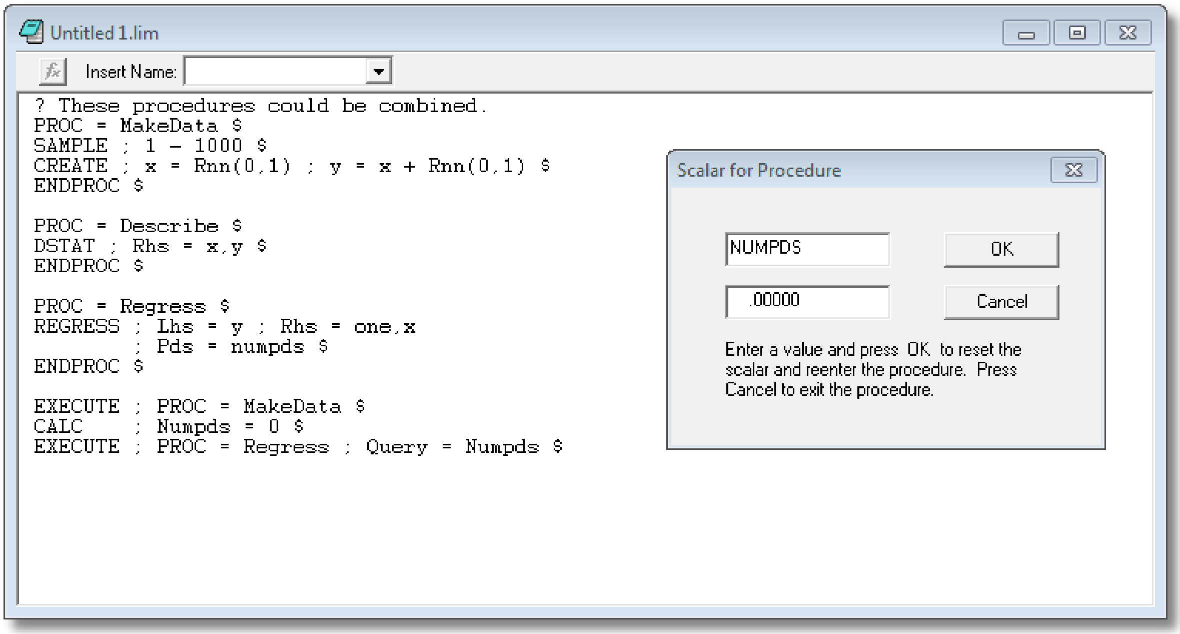
You give a value to the scalar you wish to use. Presumably this value appears in your procedure.

CALC ; name = value $

Your command is:

EXECUTE ; Query = name $

The procedure is executed first with the value set by the **CALC** command. When it is finished, you are queried for the parameter. Figure R19.9 shows operation of this feature. The number of periods for the Newey-West estimator is given by the scalar *numpds*. The value of this is set to four, then the procedure is executed. The regression is computed with *numpds* = 4 (output from the regression is not shown). After execution, a subsidiary window opens which queries you for a new value of *numpds*. The current value is shown in the window. At this point, if you wish to execute the procedure with a new value of *numpds*, you would change the value in the window and click OK. If you wish to exit the procedure, instead, you would click Cancel.



```
? These procedures could be combined.
PROC = MakeData $
SAMPLE ; 1 - 1000 $
CREATE ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
ENDPROC $

PROC = Describe $
DSTAT ; Rhs = x,y $
ENDPROC $

PROC = Regress $
REGRESS ; Lhs = y ; Rhs = one,x
        ; Pds = numpds $
ENDPROC $

EXECUTE ; PROC = MakeData $
CALC    ; Numpds = 0 $
EXECUTE ; PROC = Regress ; Query = Numpds $
```

NUMPDS
.00000

Enter a value and press OK to reset the scalar and reenter the procedure. Press Cancel to exit the procedure.

Figure R19.9 Procedure with Query for a Parameter

As another simple application, we consider a procedure to compute a ridge regression estimator,

$$\mathbf{b}_r = (\mathbf{X}'\mathbf{X} + r\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}.$$

Various ancillary computations surrounding the estimator, including the appropriate variance matrix, are discussed in Judge, et. al (1985). Here we show only the computation of the slope vector for various values of *r*. (The moment matrices are centered but not scaled.)

```
NAMELIST    ; x = list of variables ; y = Lhs variable $
CALC        ; r = 1 ; k = Col(x) $
MATRIX      ; xx = x’[1]x ; xy = x’[1]y $
PROC $
MATRIX      ; xxr = xx + r * Iden(k) ; br = < xxr> * xy $
ENDPROC $
EXECUTE     ; Query = r $
```

R19.5.6 Conditional Execution

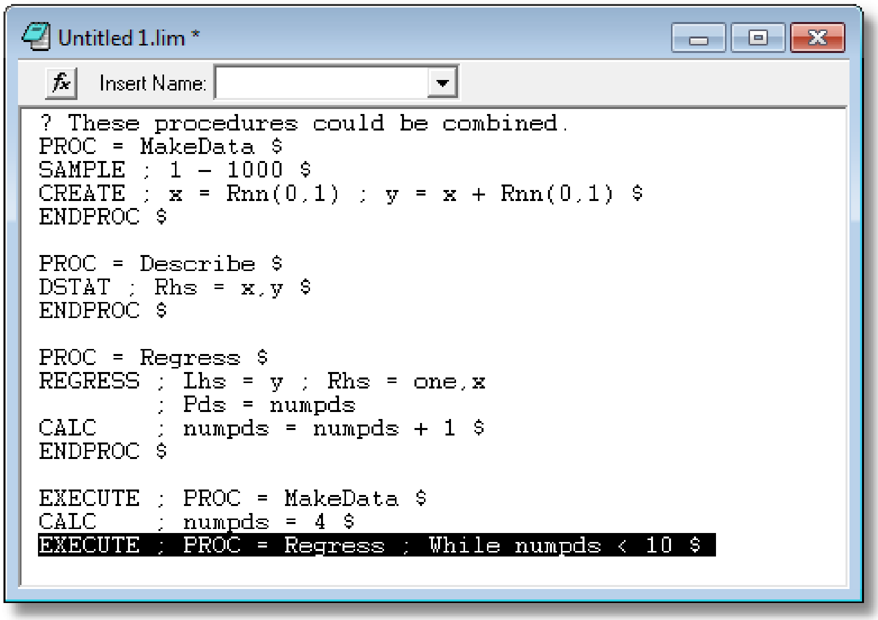
The **EXECUTE** command may be made conditional. The construction is

```
      EXECUTE     ; ... (all other options) ; While condition $
or    EXECUTE     ; ... (all other options) ; Until condition $
```

The condition is any valid condition for a *LIMDEP* logical command. (See Section R19.8.1.) For example:

```
      EXECUTE     ; Proc = integral(i) ; While i < 10 $
or    EXECUTE     ; Proc = integral(i) ; Until i*(j+r) > 1234.45  $
```

In Figure R19.10, we have modified the regression example once again, this time to request the program to compute the Newey-West estimator for several values of *numpds*. In the new procedure, the regression is computed, then *numpds* is incremented.

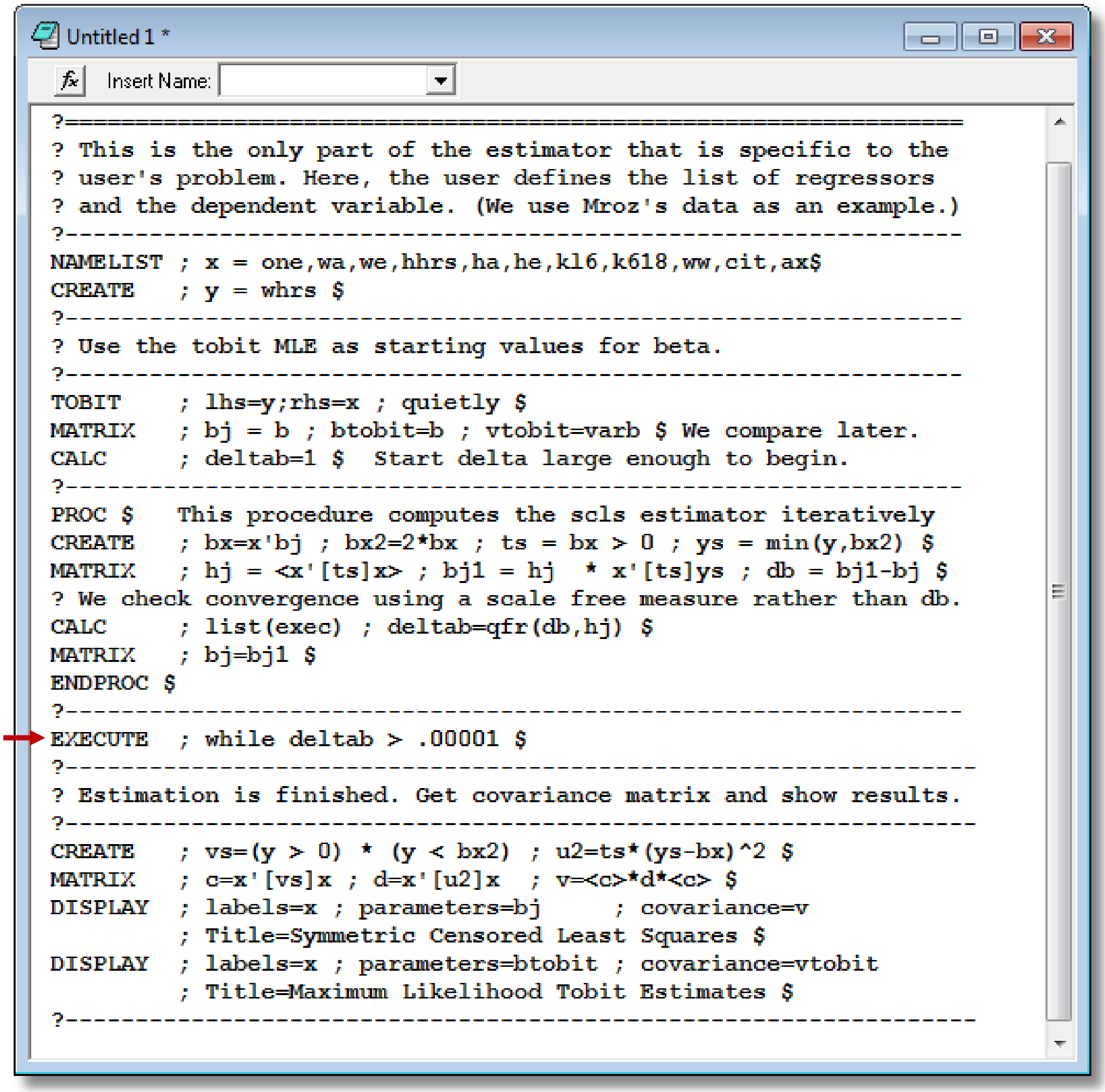


```
? These procedures could be combined.
PROC = MakeData $
SAMPLE ; 1 - 1000 $
CREATE ; x = Rnn(0,1) ; y = x + Rnn(0,1) $
ENDPROC $

PROC = Describe $
DSTAT ; Rhs = x,y $
ENDPROC $

PROC = Regress $
REGRESS ; Lhs = y ; Rhs = one,x
        ; Pds = numpds
CALC    ; numpds = numpds + 1 $
ENDPROC $

EXECUTE ; PROC = MakeData $
CALC    ; numpds = 4 $
EXECUTE ; PROC = Regress ; While numpds < 10 $
```

Figure R19.10 Conditional Execution of a Procedure

NOTE: In this example, *numpds* is set to four then the procedure is executed. As it is, *numpds* is incremented. The condition is checked before the procedure is executed. As such, for this example, the regression is computed for *numpds* = 4, 5, 6, 7, 8, and 9, but not for *numpds* = 10.

Here is a second application. *LIMDEP* does not contain a built in estimator for Powell's (1986) symmetrically trimmed censored least squares estimator. But, the estimator involves only simple least squares computations and is easily programmed using a procedure with a conditional exit rule. The program is shown in Figure R19.11. This is generic. The only changes needed for a different application are the definitions of the namelist, *x*, and variable, *y*.



```
?=================================================================
? This is the only part of the estimator that is specific to the
? user's problem. Here, the user defines the list of regressors
? and the dependent variable. (We use Mroz's data as an example.)
?-----------------------------------------------------------------
NAMELIST ; x = one,wa,we,hhrs,ha,he,kl6,k618,ww,cit,ax$
CREATE   ; y = whrs $
?-----------------------------------------------------------------
? Use the tobit MLE as starting values for beta.
?-----------------------------------------------------------------
TOBIT    ; lhs=y;rhs=x ; quietly $
MATRIX   ; bj = b ; btobit=b ; vtobit=varb $ We compare later.
CALC     ; deltab=1 $  Start delta large enough to begin.
?-----------------------------------------------------------------
PROC $   This procedure computes the scls estimator iteratively
CREATE   ; bx=x'bj ; bx2=2*bx ; ts = bx > 0 ; ys = min(y,bx2) $
MATRIX   ; hj = <x'[ts]x> ; bj1 = hj  * x'[ts]ys ; db = bj1-bj $
? We check convergence using a scale free measure rather than db.
CALC     ; list(exec) ; deltab=qfr(db,hj) $
MATRIX   ; bj=bj1 $
ENDPROC $
?-----------------------------------------------------------------
EXECUTE  ; while deltab > .00001 $
?------------------------------------------------------------------
? Estimation is finished. Get covariance matrix and show results.
?------------------------------------------------------------------
CREATE   ; vs=(y > 0) * (y < bx2) ; u2=ts*(ys-bx)^2 $
MATRIX   ; c=x'[vs]x ; d=x'[u2]x  ; v=<c>*d*<c> $
DISPLAY  ; labels=x ; parameters=bj      ; covariance=v
         ; Title=Symmetric Censored Least Squares $
DISPLAY  ; labels=x ; parameters=btobit ; covariance=vtobit
         ; Title=Maximum Likelihood Tobit Estimates $
?------------------------------------------------------------------
```

Figure R19.11 Procedure for SCLS Estimator

R19.5.7 Defining Exit (Convergence) Criteria

Procedures often involve iterations. For this sort of computation, you will need to automate the decision to continue or terminate execution. You can easily construct your own exit tests, but there are three 'test criteria' available internally. In an iterative procedure, one normally repeats a set of commands or computations, each time updating some variable(s), until some test criterion is met. For purposes of writing such a program, you can use the threshold settings in the **EXECUTE** command and compute the test values with **MATRIX** commands as follows:

1. Use **EXEC** to define up to three threshold values with the command

   ```
   EXEC     ; t(1) = value ; t(2) = value ; t(3) = value $
   ```

   Use as many of the three as desired. '*Value*' may be any fixed value or the current contents of any scalar.

2. Compute the exit values in **MATRIX** with any of

   ```
   MATRIX   ; c(j) = Norm(b1)
            ; c(j) = Chng(b1)
            ; c(j) = Chng(b1,b2)
            ; c(j) = value
   ```

   for j = 1, 2, or 3. **b1** and **b2** must be vectors. The functions shown above are

Norm	= Euclidean norm of the vector.
Chng($b1$)	= maximum of the absolute values of $b1_i$.
Chng($b1$,$b2$)	= maximum absolute value of $(b1_i - b2_i)/b1_i$, or 1000 if $b1_i = 0$.

These test criteria can be used in addition to the **; n = maximum** specification. The **EXEC** command now operates as follows:

Step 1. Execute procedure.

Step 2. If repetitions = max, exit. (Be sure to set n; the default is one, which is assumed here if you do not reset it.)

Step 3. If c(j) < t(j) for any j for which both have been defined, then exit procedure.

Step 4. Else, go to Step 1.

For an example, consider estimation of the Poisson regression model by Newton's method. The Poisson model and the necessary formulas are discussed in Chapter E41. Newton's method for the Poisson model is defined by

$$\mathbf{b}_{k+1} = \mathbf{b}_k + [\mathbf{X}'\boldsymbol{\Lambda}\mathbf{X}]^{-1}\mathbf{X}'[\mathbf{y} - \text{vec}(\boldsymbol{\Lambda})],$$

where '*k*' indexes iterations, $\lambda_i = \exp(\mathbf{b}'\mathbf{x}_i)$, $\mathbf{\Lambda}$ is the diagonal matrix of λ_is, and [$\mathbf{y}$ - vec($\mathbf{\Lambda}$)] is a vector of residuals $y_i - \lambda_i$. The matrix product is taken as the update vector for purposes of our iteration.

The following procedure and execution would estimate the model (an example is included to demonstrate the execution):

```
SAMPLE      ; 1-1000 $
CALC        ; Ran (12345) $ Set seed so you can replicate this.
CREATE      ; x1 = Rnn(0,1) ; x2 = Rnn(0,1) ; y = Rnp(4) $
NAMELIST    ; x = one,x1,x2 $
CALC        ; k = Col(x)  $
MATRIX      ; beta = Init(k,1,0) ; delta = beta $ Just to start.
PROC $
MATRIX      ; beta = beta + delta $
CREATE      ; l = Exp(x'beta) ; e = y - l $
MATRIX      ; g = x'e ; delta = <x'[l]x> * g
            ; List ; c(1) = Norm(delta) $
CALC        ; List ; Sqr(delta'delta) $
ENDPROC $
EXECUTE     ; n = 25 ; t(1) = .00001 $
```

The output is as shown below. The procedure updates the parameter vector, then computes the update vector for the next iteration. The **EXECUTE** command monitors the vector *delta*. When the norm of *delta* is small enough or 25 iterations are executed, the procedure is ended. The listing by **CALC** shows the value of the convergence rule at each iteration. The last one shown does indeed show that convergence has been reached.

```
[CALC] *Result*=      2.9938850
[CALC] *Result*=       .7995319
[CALC] *Result*=       .5546264
[CALC] *Result*=       .2257010
[CALC] *Result*=       .0303004
[CALC] *Result*=       .0005026
[CALC] *Result*=       .0000002
Exit criterion 1 met.
```

Query for Exit from Iterations

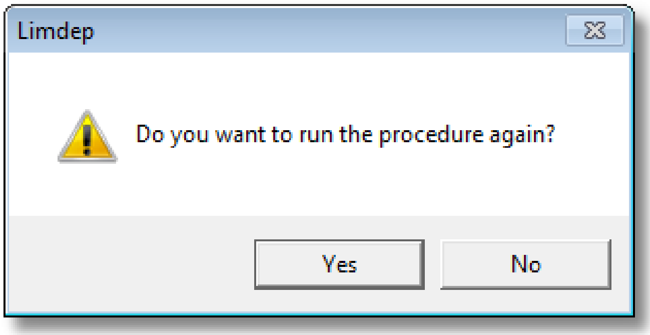
You may request simply to be queried whether to exit or not each time the procedure is carried out. In this case, *n* is ignored, so the command should be simply

```
EXECUTE     ; Query $
```

It is not necessary to set the thresholds except for convenience. Each time the procedure is carried out, the current values of *C*(1)-*C*(3) and *T*(1)-*T*(3) are displayed in your output window, and you are asked whether to reenter the iteration or to exit. Presumably the decision would rest on some previous results, including *C*(1), *C*(2), and *C*(3) if you compute them.

Figure R19.12 continues the preceding example by using **EXECUTE ; Query $** to run the procedure. The threshold values (which we have not set, so they are zero) are displayed in the output window. Then, a dialog comes up to request you to close the procedure or run it again.

```
[CALC] *Result*=      2.9938850
+------------------------------------+
|Test values for exit from procedure|
|# Current Value    Threshold Value |
|1     2.99389            .00000    |
|2       .00000           .00000    |
|3       .00000           .00000    |
+------------------------------------+
```


Limdep

Do you want to run the procedure again?

Figure R19.12 Query for Exit from a Procedure

R19.5.8 Parameters and Character Strings in Procedures

Procedures may have parameters. Define the procedure as follows:

PROC = name (parameter1, ..., up to 15 parameters) $

Then, **EXECUTE ; Proc = name (actual1, ...) $**

The actual arguments are substituted for the dummy parameters at execution time. This is like a subroutine call, but more flexible. For execution, the passed parameters are simply expanded as character strings, then the procedure, after creation in this fashion, becomes the current procedure. For example,

```
PROC = Dstats (x) $
DSTAT        ; Rhs = x $
ENDPROC $
NAMELIST     ; zbc = ... a list of variables
             ; q123 = a different list of variables $
EXEC         ; Proc = Dstats (zbc) $
EXEC         ; Proc = Dstats (q123) $
```

Any string may be substituted anywhere in the procedure. This will allow great flexibility. For example, even models can be changed with the procedure.

```
PROC = Modeler (model)
Model        ; Lhs = y ; Rhs = one,x $
ENDPROC $
EXEC         ; Proc = Modeler (probit) $
EXEC         ; Proc = Modeler (logit) $
```

Note the following for this modeler routine:

- Your procedures may have up to 15 parameters in the list.
- The number of parameters in the **EXEC** command is checked against the number in the procedure definition at execution time. But, it is not possible to check the consistency of the parameters in the two lists. Thus, you can't be prevented from sending a bad command to the **Modeler** routine above.

With this device, you are free to pass variables, namelists, matrices, model names, or any other entities you require. Also, strings can vary in type from one execution to another, though you must be careful to avoid causing conflicts. For example, assuming that *t* is a scalar in the named procedure, one might use

```
PROC = name (t) $
```

then,

```
EXEC         ; Proc = name (x) $
EXEC         ; Proc = name (1.2345) $
```

which would not cause a conflict.

Such procedures would generally be useful for creating prepackaged subroutines. For example, the following procedure computes LM statistics for a given model using two sets of variables:

```
PROC = Lmtest (model, y, x1, x2) $
Model        ; Lhs = y ; Rhs = x1 $
MATRIX       ; k2 = Col(x2)
             ; b2 = k2 _ 0 $
Model        ; Lhs = y ; Rhs = x1,x2
             ; Start = b, b2 ; Maxit = 0 $
ENDPROC $
```

You could execute this with something like the following:

```
NAMELIST     ; v1 = one,v1a,ddd
             ; v2 = ll,g123 $
EXEC         ; Proc = Lmtest (probit, y, v1, v2) $
```

Macros – The STRING Command

The **STRING** command provides another way for you to set up variable parameters in your procedures. This feature allows you to define a character string with a name. This may save you some typing. But, you can also use this command to operate your procedures more like true subroutines. *LIMDEP* keeps three cells for you to use to store character strings with up to 80 characters each. The command to store a character string is

```
STRING      ; stj = any character string $
```

where ‘*j*’ is 0, 1, or 2. The only rule that applies is that the string may not be used to give the primary verb of a command or the semicolon which follows it. For example,

```
STRING      ; st1 = dstat ; Rhs = * $
```

is not a valid command, because **DSTAT** is a primary verb (model name). But, you might use

```
STRING      ; st1 =  Rhs = * ; Output = 3 $
```

The string is inserted into a command by enclosing its name in double quotes. To continue the example, following the preceding, you could use

```
DSTAT       ; “st1” $
```

Strings can make procedures work like subroutines, as in

```
PROC $
DSTAT       ; “st0” $
ENDPROC $
STRING      ; st0 = Rhs = c $
EXECUTE
STRING      ; st0 = Rhs = y $
EXECUTE
```

This is essentially the same as using a parameter list as shown at the beginning of this section. The advantage would be for cases in which you want to modify longer character strings, rather than just names or values.

R19.5.9 Local Variables in Procedures

Your procedures may contain any commands and use any variables, matrices, etc. that exist in your project. One implication of this flexibility is that to this point, all entities that you use in your procedures are ‘global.’ What you compute within your procedure affects your project after the procedure is completed. For example, if your procedure contains the command **CALC ; rho =.7 $**, when the procedure is completed, the value of the scalar *rho* in your project will equal .7. You may be interested in creating scalars, matrices or variables that are ‘local’ to your procedure. These would typically be intermediate computations that you are not interested in retaining after the procedure has been executed.

You can declare local variables in a procedure as follows:

```
PROCEDURE $
LOCAL       ; Scalar   = a set of names for the scalars you want to use
            ; Matrix   = a set of matrix names for matrices you want to use
            ; Variable = a set of variable names that you wish to use $
… commands that use these names and any others …
ENDPROCEDURE $
EXECUTE     ; … any options $
```

You may use any or all of the three type declarations in the **LOCAL** command. The entities that they define will exist while the procedure is being executed, but they will disappear after the procedure is finished.

Note the following parameters for the **LOCAL** command:

- The names that you declare may be new names that do not already exist in the project. In this case, these scalars, matrices or variables will vanish after the procedure is executed.

- The names that you declare may be the same as entities that already exist in the project. In this case, the local entity will use that name temporarily, but when the procedure is finished, the existing entity is restored. For example, this procedure

  ```
  CALC      ; List ; a1 = pi $
  PROCEDURE $
  LOCAL     ; Scalar = a1 $
  CALC      ; List ; a1 = Sqr(pi) $
  ENDPROCEDURE $
  EXECUTE $
  CALC      ; List ; a1 $
  ```

  sets the scalar $a1$ equal to π. Within the procedure, $a1$ is set equal to the square root of π. When the procedure is finished, $a1$ is still equal to its global value, π. The visible results of this procedure are

  ```
  [CALC] A1      =      3.1415927
  [CALC] A1      =      1.7724539
  [CALC] A1      =      3.1415927.
  ```

- The procedure may also use global names. If they are not declared to be local to the procedure, then scalars, matrices and variables that are created within the procedure will exist when the procedure is finished.

R19.6 Looping with the EXECUTE Command

The **EXECUTE** command may be set up in the form of a 'DO LOOP.' The syntax would be

```
EXECUTE    ; name = first value, last value, increment ; ... other options $
```

The command works as follows: When you give this command, 'name' is created as a scalar in your calculator work area. It may already exist or you can create a new scalar this way. The loop parameters may be any values, real or integer, positive or negative. For example,

```
           ; rho1 = 0 , 1, .1
```

creates scalar *rho*1 and moves it from zero to one in steps of 0.1. In this case, the procedure would be executed 11 times. You can also decrement, as in

```
           ; index = 100.781, 20.11, -21.5,
```

which produces five repetitions. The increment may be omitted, in which case, it is assumed to be one. Thus, for example,

```
EXECUTE    ; i = 1, 10 $
```

Before executing, *LIMDEP* determines the number of repetitions using

$$\textit{repeat count} = \text{Min}(1, |\textit{last} - \textit{first}| / |\textit{increment}|)$$

so the procedure is always executed at least once. This setup is similar to

```
EXECUTE    ; n = repeat $
```

except that the looping form creates a scalar entity which you can use in your calculations.

The loop index is a scalar which you may use in other commands. But, you may not change it with any other command. It is 'read only' while the loop is executing. Thereafter, you may change it in any way you like.

R19.7 Looping Over an Indexed Set

R19.7.1 Looping Over a Set of Variables

Variables in namelists may be indexed. The format is *listname:index* to indicate the '*index*th' variable in the list. For example, in

```
NAMELIST   ; x = yabc,ydef,y123 $
```

x:1 is *yabc*, *x*:2 is *ydef*, and, *x*:3 is *y*123. This indexing scheme can be used in any command at any point in a command stream. It is most likely to be useful in a procedure, however.

For example, the following procedure takes the logs of up to 100 variables:

```
NAMELIST    ; x = ... $
PROC $
CREATE      ; z : i = Exp(x : i) $
DSTAT       ; Rhs = x : i, z : i $
ENDPROC $
CALC        ; x = Col(x) $
EXEC        ; i = 1,k $
```

A list of variable names may also be used to control execution of a block of commands, as in the block below.

PROCEDURE

Commands use variable : *loopname*.

```
ENDPROC
EXECUTE     ; : loopname = a list of variables $
```

This could be useful, for example, for estimating the same model with a set of dependent variables. For example:

```
PROC $
REGRESS     ; Lhs = : shares ; Rhs = x $
ENDPROC $
EXECUTE     ; : shares = capital,labor,fuel $
```

R19.7.2 Looping over Observation Tags

Your data may be stratified with alphabetic observation tags (see Section R3.2.2). You may use these to control iteration of a procedure. The general syntax is

```
EXECUTE     ; For [ {tagname} = list of tags ] ; … $
```

In the example developed in Chapter R3, the 816 observations are grouped into 48 states with tags ST_ABB = AL, AZ, AR, etc. There are also 11 regions defined by region tags including MW, MA, NE, and so on. You can use these labels, such as

```
EXECUTE     ; For [ {REGION} = MW,MA,NE ] $
```

If the “= list” is omitted, then execution is over all defined tags. Thus,

```
EXECUTE     ; For [ {ST_ABB} ]  ; … $
```

would produce 48 iterations of the procedure, one for each state. During the iteration, the active sample is the set of observations that have that observation tag.

R19.8 Flow Control within Procedures

The ability to construct loops is one of the most important features of any programming language. The next two sections will give a large amount of detail on how to construct loops and iterative programs. In this section, we focus on one set of commands and techniques. The **LABEL** and **GO TO** commands, which we used in Section R19.4, allow the creation of multiple and nested loops. Programs may contain multiple labels and several loops, which may be nested or executed one after the other.

The basic construction is:

```
PROCEDURE $
... some block of commands ...
LABEL         ; number $
... some other block of commands ...
GO TO         ; number ; logical condition $
ENDPROCEDURE $
EXECUTE
```

The **GO TO** instruction is carried out if the logical condition is true. If no condition is given, then the **GO TO** is unconditional – it is always carried out.

R19.8.1 Logical Expressions

The logical condition can relate any scalar entity to any other scalar entity (i.e., scalars, numbers, or matrix elements) and can be as complex as needed to carry out the desired task. The structure to use for this part of the command is the same as that for **CREATE** (Section R4.2.2), **CALC** (Section R17.4), **MATRIX** (Section R16.2.3), and **REJECT/INCLUDE** (Section R7.4). Logical expressions are any desired expressions that provide the condition for the transfer of control to be carried out (or not). They may include any number of levels of parentheses and may involve mathematical expressions of any complexity involving named scalars, matrix or vector elements, and literal numbers. The operators are:

Math and relational:	+, -, *, /, ^, >, >=, <, <=, =, #.
Concatenation:	& for 'and', | for 'or.'

A simple example is:

```
GO TO         ; 100 ;  x > 0 $
```

For a second example with no obvious interpretation:

```
GO TO         ; 100 ;  (r / s)*((c + 7)*(x + 2) * y^2 + z^3) > 1  |  x + y < 0 $
```

The hierarchy of operations is ^, (*,/) (+,-), (>,>=,<,<=,=,#), &, |. Operators in parentheses have equal precedence and are evaluated from left to right. When in doubt, add parentheses. There is essentially no limit to the number of levels of parentheses. (They can be nested to about 20 levels.)

It is important to note that in evaluating expressions, you get a logical result, not a mathematical one. The result is either 'true' or 'false.' An expression which cannot be computed cannot be true, so it is false. Therefore, any subexpression which involves division by zero or a negative number to a noninteger power produces a result of false. But, that does not mean that the full expression is false. For example: '*x* / 0>0 | *x*>*y*' could be true. The first expression is false because of the zero divide, but the second might be true, and the OR in the middle returns true if either expression is true. Also, we adopt the C++ language convention for evaluation of the truth of a mathematical expression. A nonzero result is true, a zero result is false. Thus, your expression need not actually make logical comparisons. For example: Suppose *x* is a scalar which might be 0 or 1. **GO TO ; 100 ; x $** will make the transfer if *x* equals 1 and not if *x* equals 0. Therefore, this is the same as **GO TO ; 100 ; x # 0 $**.

NOTE: Using variables in the logical expressions is permissible. But, except for **CREATE**, **REJECT**, and **INCLUDE**, which are being evaluated during a loop through your data set, the values taken by variables will be ambiguous, and the results will be unpredictable. For example, if *x* is a variable, **GO TO ; 974 ; x > 1 $** is likely to behave strangely since the value taken by *x* is generally defined only by the last operation to use your data.

R19.8.2 Loops within Procedures

There are two commands which can be used to transfer control in procedures and in editor files. These are

```
        LABEL          ;  label number $
and     GO TO          ;  label ; condition $
```

As discussed earlier, these commands can be used to add flexibility to procedures. One usage would be to create loops within procedures, or even loops within loops. The following example (rather clumsily) computes recursive residuals for a set of observations. The procedure itself would be executed once, but the loop within the procedure is controlled by the **LABEL** and **GO TO** commands.

```
NAMELIST    ; x = ... $
PROC = recrsive $
CREATE      ; recrsiv = 0 $
CALC        ; i = k $
LABEL       ; 100 $
SAMPLE      ; 1-i $
MATRIX      ; bt = Xlsq(x,y) $
CALC        ; i = i + 1
SAMPLE      ; i $
CREATE      ; recrsiv = y - x'bet $
GO TO       ; 100 ; i <= N $
ENDPROC $
```

This routine computes a set of recursive residuals for observations *k*+1 to *n* in a sample. Note that the statements which control the loop are the initial value of *i*, the incrementing of *i* and the **GO TO** statement.

Examples

Consider the following example:

```
PROC $
CREATE      ; recrsiv = 0 $
CALC        ; i = 2
            ; nobs = n $
LABEL       ; 100 $
SAMPLE      ; 1-i $
MATRIX      ; bt = Xlsq(one,x,y) $
CALC        ; i = i +1
            ; laste = y(i) - bt(1) - bt(2)*x(i) $
CREATE      ; recrsiv(i) = laste $
GO TO       ; 100 ; i  <= nobs $
ENDPROC $
EXECUTE
```

This routine computes a set of recursive residuals for observations 3 **-** *n* in a sample. Note again that the statements which control the loop are the initial value of *i*, the incrementing of *i* and the **GO TO** statement.

As shown above, the loop could have been controlled with a single **EXECUTE** command. But, this new structure allows much greater flexibility. First, the transfer may be forward or backward in the command block. Thus, one can execute any block of commands conditionally, by bypassing it or not depending on a condition. Second, there may be any number of transfers (up to 10 altogether) in the code block, and these can be nested. For example:

```
PROC $
CALC        ; i = 0 $
LABEL       ; 10 $
CALC        ; i = i + 1 $
CALC        ; j = 0 $
LABEL       ; 20 $
CALC        ; j = j + 1 $
MATRIX      ; matrix (i,j) = pi $
GO TO       ; 20 ; j  < 5 $
GO TO       ; 10 ; i  < 5 $
ENDPROC $
EXECUTE
```

This loads a 5×5 matrix with π one cell at a time (perhaps a bit clumsily, since **MATRIX ; matrix = Init(5,5,pi) $** does the identical job).

Although our examples have been loop constructions, the **GO TO** command can operate on any scalar comparison. Thus, one might compute a regression, then do some followup computations if the R^2 is greater than .5, and so on.

R19.9 Looping with DO Statements

You may use four forms of **DO** statements as alternatives to the **LABEL** and **GO TO** commands in the editor. The **DO** command is used to request repetition of a block of commands. It has four forms. The first is:

```
PROC $
DO FOR        ; label ; index = a1[,a2,[a3]] $
  ... any block of commands ...
ENDDO         ; label $
ENDPROC $
EXECUTE
```

The label is any character string, name, or number, up to eight characters. This is carried out as follows:

- If only *a*1 is given, the block of commands is carried out once. For example,

```
DO FOR    ; Ar1model ; rho = .5 $
...
ENDDO     ; Ar1model $
```

- If *a*1 and *a*2 are given, the block of commands is carried out once for each of the values *a*1, *a*1+1, *a*1+2, ..., until the incremented value is greater than or equal to *a*2. Typically, these would be integers, as in

```
MATRIX  ; a = Init(10,1,0) $
DO FOR  ; 10 ; i = 1,10 $
MATRIX  ; a(i) = i $
ENDDO   ; 10 $
```

If *a*3 is given, the execution is the same as in the preceding example except *a*3 is used as the increment instead of 1. For example, the following computes the Box-Cox model for *lambda* = -1, -.9, ..., .9, 1.0.

```
DO FOR  ; bc ; m = -1,1,.1 $
BOXCOX  ; ... ; lambda = m $
ENDDO   ; bc $
```

Three other forms of the **DO** command are as follows:

- Do while executes the block of commands as long as the condition is true.

```
DO WHILE ; label ; logical condition $
...
ENDDO    ; label $
```

The block of commands is always executed once. Then the condition is checked and if true, the commands are executed again. There is potential for problems here. If your block of commands does not change something that can falsify the condition, this block of commands will execute forever.

- Do until executes the block of commands until the condition becomes true.

```
DO UNTIL ; label ; logical condition $
...
ENDDO    ; label $
```

- Do if executes the block of commands once if the condition is true.

```
DO IF    ; label ; condition $
...
LABEL    ; label $
```

The logical condition in these constructions is set up the same as for the **GO TO** instruction. See Section R19.8.1 for details. For example:

```
CALC         ; q = 0 $
DO UNTIL     ; 10 ; q > .5 $
CALC         ; q = ... some operation that increases q $
ENDDO        ; 10 $
```

You can nest loops up to 10 levels. For example:

```
PROCEDURE $
MATRIX       ; a = Init (3,3,0) $
DO FOR       ; 10 ; i = 1,3 $
DO FOR       ;  5 ; j = 1,3 $
CALC         ; ij = i * j $
MATRIX       ; a(i,j) = ij  $
ENDDO        ;  5 $
ENDDO        ; 10 $
ENDPROCEDURE $
```

There is a limit of 10 loops in one command block, but these may be arranged in any fashion you like.

WARNING: There are many ways to construct bad loops, and by and large, *LIMDEP* cannot protect you from them. For instance, in the previous example, if the second to last and the last statements were reversed, the loop would be badly nested. The results might not be what you expected. Also (note carefully!!), it is possible for you to create infinite loops from which there is no escape. For example, if you decrement instead of increment a counter, there may be no way to get out of a loop. Unfortunately, it is usually not possible for *LIMDEP* to check such a condition just by looking at a set of commands, so it is solely up to you to avoid this. The next section discusses what to do in such a situation.

The **DO FOR** command must appear within a procedure. It can be used to set up looping procedures based on counters, as in C++ or Fortran. The general syntax is

```
DO FOR       ; label ; index = the specification $
```

This command is set up so that loop counters are dynamically computed. This provides the ability to nest loops, and set the counter for an inner loop in an outer one. For example,

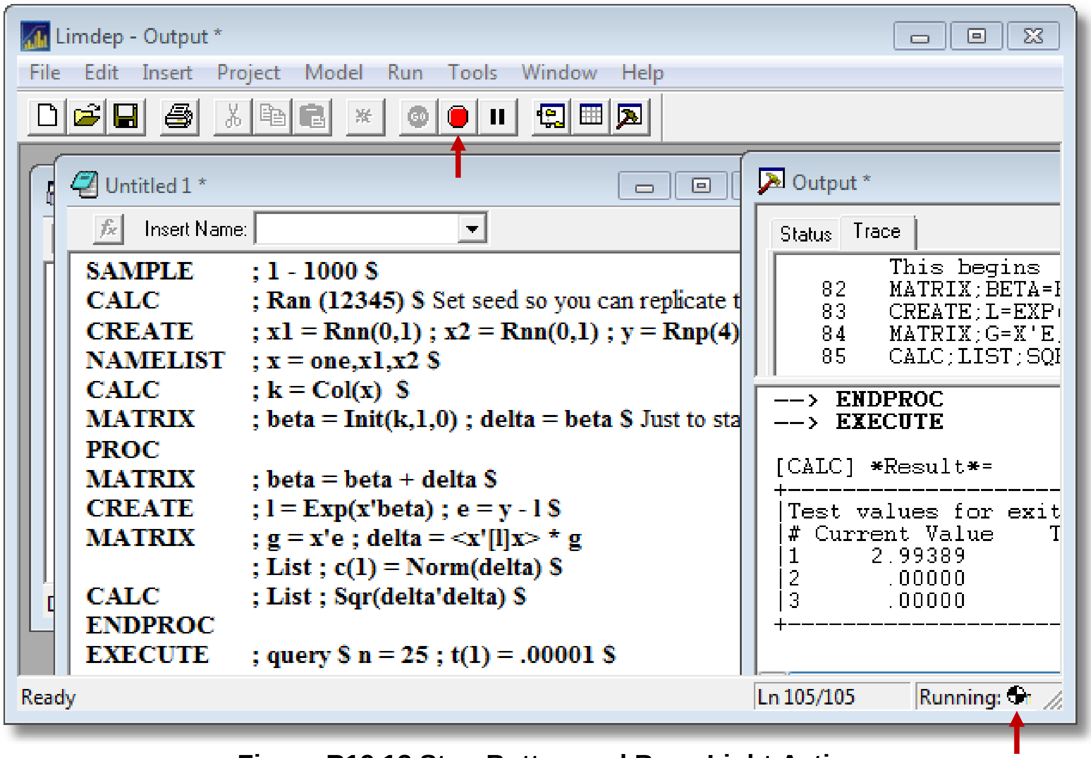
```
DO FOR      ; 10 ; k = 1,10 $
DO FOR      ; 5 ; j = 1,k $
CALC        ; sum = j+k $
MATRIX      ; m(j,k) = sum $
ENDDO       ; 5 $
ENDDO       ; 10 $
```

This block of lines places values in the upper triangle of a matrix. The lower triangle is unchanged. (Note, it might be tempting in the procedure to just use **MATRIX ; m(j,k) = j+k $**, however, the Rhs of this command is not a matrix computation, so it will not be carried out correctly.)

R19.10 Escaping from an Infinitely Looping Procedure

The warning about infinite loops in the preceding section suggests an aspect of operation of *LIMDEP* that it is useful to know a bit about. Much of the kind of computation that you do with *LIMDEP* involves mathematical calculations that take place in the background, particularly if you are using large data sets or simulation based estimators. As such, there may be long intervals when you are waiting passively for results while the program computes. Unfortunately, the infinite loop scenario suggested above would be one of these situations. *LIMDEP* has two ways to tell you when computation is going on in the background. First, the Stop button on the *LIMDEP* toolbar will be illuminated red, as shown in Figure R19.13. (At other times, it will be dark.) Second, there is a 'busy light' (the black and white spinner at the lower right corner of the desktop) which appears during a computation. If you have created an infinite loop, or some other inescapable situation, the busy light will be spinning, but you will be unable to use the Stop button to exit the computation. There is no recourse but to exit the program.

If you find yourself having to 'crash' *LIMDEP*, or any other program for that matter, there is a tool in Windows that can do this fairly painlessly. When you need to escape from any program and have no other recourse, place your mouse cursor in an empty space in the taskbar at the bottom of your screen and right click. From the popup menu, select Start Task Manager. From the Applications tab, you can select *LIMDEP* from the list of running applications and then select End Task to get out of your infinite loop. Unfortunately, as the menu indicates, this terminates the program. However, it is a polite termination. Before the program ends, you will be given an opportunity to save your project and editing windows.



```
SAMPLE      ; 1 - 1000 $
CALC        ; Ran (12345) $ Set seed so you can replicate t
CREATE      ; x1 = Rnn(0,1) ; x2 = Rnn(0,1) ; y = Rnp(4)
NAMELIST    ; x = one,x1,x2 $
CALC        ; k = Col(x)  $
MATRIX      ; beta = Init(k,1,0) ; delta = beta $ Just to sta
PROC
MATRIX      ; beta = beta + delta $
CREATE      ; l = Exp(x'beta) ; e = y - l $
MATRIX      ; g = x'e ; delta = <x'[l]x> * g
            ; List ; c(l) = Norm(delta) $
CALC        ; List ; Sqr(delta'delta) $
ENDPROC
EXECUTE     ; query $ n = 25 ; t(l) = .00001 $
```

```
           This begins
  82       MATRIX;BETA=B
  83       CREATE;L=EXP(
  84       MATRIX;G=X'E.
  85       CALC;LIST;SQR

--> ENDPROC
--> EXECUTE

[CALC] *Result*=
+------------------
|Test values for exit
|# Current Value    T
|1      2.99389
|2       .00000
|3       .00000
+------------------
```

Figure R19.13 Stop Button and Busy Light Active

R19.11 Editing Procedures and Creating New Procedures

Once you have entered a procedure, there are two ways you can edit it, for example, if it is necessary to correct errors.

1. If the procedure is in the text editing window, the easiest way to proceed is just to edit it on the screen, then highlight just the procedure, from **PROC** through **ENDPROC**, and click the GO button on the toolbar. This will replace the procedure with the modified version.

2. If you have one or more named procedures, you can also use *LIMDEP*'s procedure editor to edit them. Open the Procedures folder in the project window, then double click the name of the procedure. This will open the procedure editing window.

To create a new procedure, use any of the following:

1. Type your **PROCEDURE** commands in the editing window, highlight the procedure from **PROC** to **ENDPROC** and click GO.

2. From the Project menu, select New, then Procedure.

3. From the Run menu, select New Procedure.

4. From the Insert menu, select Item into Project, then Procedure.

The last three options above all open the New Procedure dialog box, shown in Figure R19.14.

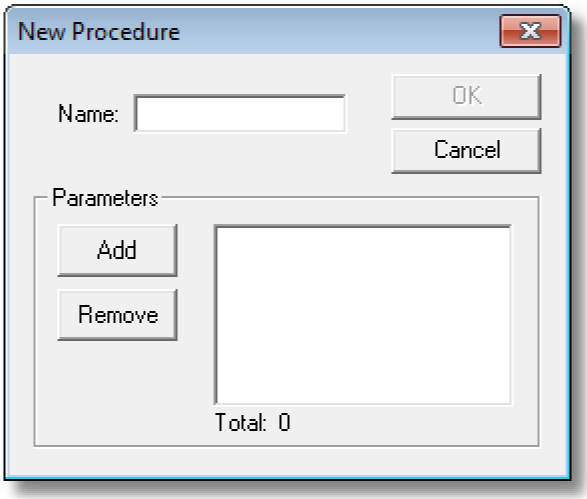
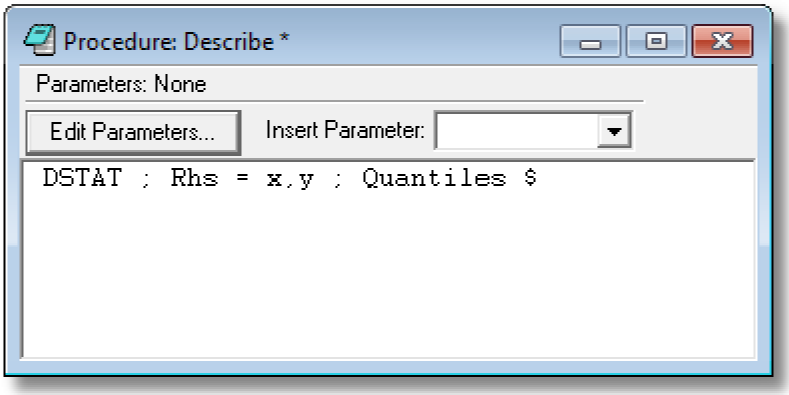


Figure R19.14 New Procedure Dialog Box

You can then enter the procedure name and any parameters you wish. This sets up the header for the procedure. When you exit with OK, the editing window will then appear as shown in Figure R19.15.



```
DSTAT ; Rhs = x,y ; Quantiles $
```

Figure R19.15 Procedure Editing Window

(You will go directly to this window if you are editing an existing procedure.) Within the procedure editing window, you can enter the commands, change existing commands, or modify the parameter list. The standard features of the Edit menu (cut, paste, etc.) are all available in this window as well. Upon exiting, the procedure will be entered into the library.

R20: Multiple Imputation

R20.1 Introduction to Multiple Imputation

Multiple imputation (MI) is a set of techniques that is used to fill missing values in a data set when estimating a model. The core of the technique is an ancillary model that is used to fit the missing values prior to estimation of the model of interest. To form the basic framework, consider a model

$$f(y|\mathbf{x},\boldsymbol{\beta},\boldsymbol{\theta}) \;=\; g(y,\mathbf{x},\boldsymbol{\beta},\boldsymbol{\theta}).$$

For example, $f(y|\mathbf{x},\boldsymbol{\beta},\boldsymbol{\theta})$ might be a normal density with conditional mean function $\boldsymbol{\beta}'\mathbf{x}$ and conditional variance $\theta = \sigma^2$. Suppose estimation is to be based on N observations, however, n_m observations on $\mathbf{x}$ are missing (and n_c observations are complete). The technique works in three steps:

Step 1. Fit imputation equation $\hat{x} = h$ (available sample information) using complete data.

Step 2. (Repeated M times). Use the imputation equation to fill the missing values of $\mathbf{x}$, then estimate $(\boldsymbol{\beta},\boldsymbol{\theta})$ using imputed sample m.

Step 3. Aggregate the M estimates of $(\boldsymbol{\beta},\boldsymbol{\theta})$ and the M estimates of the asymptotic covariance matrix for the estimates.

Step 1 is preparatory. Step 2 consists of M repetitions of an ‘imputation step’ followed by an ‘estimation step.’ The imputation step is based on a Bayesian approach to obtaining an appropriate random sample to use to fill the randomly missing observations.

The reader is referred to Rubin (1976, 1987, 1996), Little and Rubin (2002), and Royston (2004) for some details. There is a large intricate literature on the fine details and various aspects of MI that apply when the data do not conform to the multivariate normal conditions that are ideal for the techniques. *LIMDEP* provides a basic set of procedures for MI for a variety of types of data. Among the advantages of this implementation:

- An enormous disadvantage of some implementations of MI is the need to replicate the data set for each iteration. If the base data set is large, this limits the imputation to only a few repetitions. *LIMDEP* avoids this problem by storing only the formulas needed to do the imputation, not the imputed data themselves. Imputations are created within the existing data set, not by replicating the data. This implies that there is no practical limit to the number of iterations that can be performed at Step 2.

- As a consequence of the first advantage, there is no list of specific procedures that MI can be used with in *LIMDEP*. MI is available for *every* procedure that uses data in *LIMDEP* including estimators that you create with **MAXIMIZE** or with any other procedures. That is, Step 2 above is not estimation of a specific model; it is one or more (possibly many more) computations using the data set that contains imputed values.

R20.2 Methodology

The template application of MI can be drawn with reference to a model

$$y = f(x1,x2|\boldsymbol{\beta})$$

where $\boldsymbol{\beta}$ is the parameter vector to be estimated. We suppose that there are n observations in the sample, $n_{c,1}$ complete observations on $x1$, $n_{m,1}$ missing values for $x1$, and $n_{c,2}$ and $n_{m,2}$ complete and missing observations on $x2$. The missing and complete observations on $x1$ and $x2$ need not coincide. We suppose as well that there is additional information in the sample, **Z**, for which there are observations present for at least some observations when there are missing observations on $x1$ or $x2$. (Though there is mention of it in some of the received literature, we will ignore the case of missing values on y. Prediction of missing values of the dependent variable in an equation could not possibly pass the 'missing at random' test needed to use MI and could not be exogenous in the model as is required for consistent parameter estimation.) The overall approach of MI is to use available information on $x2$ and **Z** to predict missing values of $x1$ and available information on $x1$ and **Z** to predict missing values of $x2$. It is assumed that the missing values are 'missing at random,' that is, that the data on $x2$ and **Z** do not contain information on the probability that $x1$ is missing, and likewise for $x1$ and **Z** for $x2$. The three steps listed above are carried out as follows:

Step 1. Construct imputation equations $\hat{x}1 = h_1(x2, Z, \hat{\delta}_1)$ and $\hat{x}2 = h_2(x1, Z, \hat{\delta}_2)$ using available complete observations on relevant variables.

Step 2. (M repetitions): Simulate missing values of $x1$ from the conditional model h_1 and missing values of x_2 from the conditional model h_2. For each repetition, we obtain estimates of the parameters, $\hat{\boldsymbol{\beta}}_m$ and the asymptotic covariance matrix $\hat{\boldsymbol{\Sigma}}_m$.

Step 3. (Aggregation). The estimator of $\boldsymbol{\beta}$ is $\bar{\mathbf{b}} = \frac{1}{M}\sum_{m=1}^{M}\hat{\boldsymbol{\beta}}_m$. The variance estimator is

$$\bar{\mathbf{S}} = \frac{1}{M}\sum_{m=1}^{M}\hat{\boldsymbol{\Sigma}}_m + \left(1+\frac{1}{M}\right)\frac{1}{M-1}\sum_{m=1}^{M}\left(\hat{\boldsymbol{\beta}}_m - \bar{\mathbf{b}}\right)\left(\hat{\boldsymbol{\beta}}_m - \bar{\mathbf{b}}\right)'$$

Steps 1 and 3 are straightforward. The complication at Step 1 is the choice of imputation equation. The most thoroughly researched case is the linear regression model, which would be used if $x1$ or $x2$ were a continuous random variables without obvious other complications. However, survey data and large public data sets typically are composed mostly of binary or attitudinal scale variables for which linear regression methods are inappropriate. We note in the next section the several types of models that *LIMDEP* uses for missing data situations. Step 3 obtains the estimator of the asymptotic variance of the MI estimator. The variance estimator accounts for the essential sampling variability of the estimator in the first term – this is the 'within simulation' variance – and the variability introduced by the simulation itself in the second term – this is the between simulations variance. Step 2 is the focus of attention in the received literature. There are many approaches used for the simulation. The description below details the method used in *LIMDEP*.

R20.3 How It's Done – Overview

The implementation of MI in *LIMDEP* proceeds as follows:

For Step 1, you will fit as many as 30 equations for variables that contain missing values that you intend to impute during your analysis. For an example, we suppose we will impute missing values for *doctor*, which is binary and for *hhninc* (household income) which is continuous. This step proceeds as follows:

```
IMPUTE ; Lhs = doctor ; Rhs = one,age,educ,married ; Type = binary $
----------------------------------------------------------
Equation stored for imputing missing values of     DOCTOR
Imputation method: Binary Logistic
Observations currently in full data set        =    27326
Complete observations for imputation equation  =    17897
Missing observations on   DOCTOR in data set   =     2765
----------------------------------------------------------
```

```
IMPUTE ; Lhs = hhninc ; Rhs = one,age,educ,handper ; Type = measurement $
----------------------------------------------------------
Equation stored for imputing missing values of     HHNINC
Imputation method: Linear Regression
Observations currently in full data set        =    27326
Complete observations for imputation equation  =    17768
Missing observations on   HHNINC in data set   =     2839
----------------------------------------------------------
```

The project window is updated to include the information about the imputation equations.

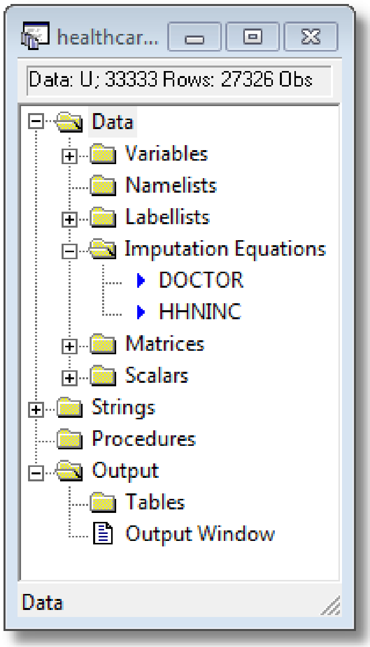


Figure R20.1 Imputation Equations in the Project Window

By double clicking either of the imputation equations in the project list, we produce a summary table of the results of the estimation step for the imputation equations.

```
IMPUTE ; List $
-------------------------------------------------------------
Imputation Equations
Equations currently available for multiple imputation
The full sample contains       27326 observations
Complete observations counts apply to full sample
Missing obs for RHS is based on listwise deletion
Types: LR=linear regression, BR=binary logistic
       OL=ordered logit,     ML=multinomial logit
       FL=fractional logit,  PR=Poisson regression
---------------------+--------------+--------------------+
      Equation       |    RHS Vars  |Complete Observations|
LHS       type missing|Number Missing|   LHS    RHS     Eqn|
---------------------+--------------+--------------------+
DOCTOR    BR     2037|   4      7392| 25289  19934   17897|
HHNINC    LR     2090|   4      7468| 25236  19858   17768|
---------------------+--------------+--------------------+
```

Step 2 is carried out by defining a procedure as described in Chapter R19. The template for the procedure appears as in the following example:

```
PROC $
PROBIT        ; Lhs = ... ; Rhs = ... x... ; Imputation = imputna $
LOGIT         ; Lhs = ... ; Rhs = ... x... ; Imputation = imputnb $
POISSON       ; Lhs = ... ; Rhs = ... x... ; Imputation = imputnc $
ENDPROC $
EXECUTE       ; N = number of imputations desired
              ; Imputation = imputna,imputnb,imputnc $
```

The procedure acts as follows: Each model for which we wish to use the MI procedure is given a name – there are three in the procedure. The **EXECUTE** command instructs *LIMDEP* to fit each model *N* times with imputed data each time. For example, in the preceding, if we set *N* = 5, then there would be five different imputed data sets used to fit the models. A new data set is created for each iteration. Note, however, that the imputation is independent of the model. It is carried out before the model is estimated. For example, the first model command in the procedure is a **PROBIT** equation. Since this is one of the imputation models, before the probit model is fit, the imputation equations are used to fill as many observations as possible. After each repetition of the procedure, the data set is restored to its original state, with the missing values back in place. Then, the steps are repeated for each imputation. An important implication of this sequencing of the steps is that if the model equation contains interactions or nonlinearities, these will be applied correctly to the imputed data. That is, within the procedure, the sample data set, for any and all purposes, is the imputed data set for that repetition.

R20.4 The Imputation Step

This section describes creating the imputation equations. Though we label this the imputation step, in fact, the imputation takes place at the same time as the estimation is done. Your first step in this analysis is to create the equations used to compute the imputed values. Do this as follows:

1. Use **SAMPLE ; All $** to use as much information as is in the data as possible.
2. For each variable that you intend to fill with missing values, create the imputation equation with

```
IMPUTE ; Lhs = the variable
       ; Rhs = one,… the variables that you will use for the imputations
       ; Type = the type of variable on the Lhs $
```

The types are

M = measurement – continuous variable, use linear regression
B = binary variable, use a logit model to predict
C = count variable, use a Poisson regression
O = ordered (scale) variable, use an ordered probit model
F = fractional variable, use a logit model for proportions data
T = type variable – unordered choice, use a multinomial logit model

There are no other optional specifications for the **IMPUTE** command. This instruction fits a model using as many complete observations as it can find in the current sample. No estimation results are produced. A summary count of the number of complete and incomplete observations that were encountered is produced. An example appears in Section R20.3. You may store up to 30 equations each with up to 100 Rhs variables including the constant term. The list of imputation equations, identified by the Lhs variables, appears in the project window. You can inspect the results of this step by double clicking any of the names in the project window.

To delete an imputation equation, use

```
IMPUTE ; Delete name $
```

(There is no semicolon before the name.) You can also delete an equation by right clicking the name in the project window and selecting Delete from the menu. Obtain a summary of the set of imputation equations with

```
IMPUTE ; List $
```

You can use your imputation equations to create simulated values for the missing values in your data set as follows: First, create a template for the simulation with

```
CREATE ; new variable name = old variable name $
```

The new variable name is any name you wish to use for the variable to be created. The old variable is a Lhs variable whose name appears in the list of imputation equations. Second,

```
IMPUTE ; Lhs = new variable ; Rhs = old variable ; Type = Fill $
```

LIMDEP will use the stored imputation equation and the method described in Section R20.7 to fill as many missing values as possible.

TIP: Neither the theory nor *LIMDEP*'s implementation of it will prevent you from using **IMPUTE** to create imputed values of the dependent variable in an equation. The theory should prevent this, however, since the imputed values will be endogenous in the resulting equation. Any notion of consistent estimation of the model parameters would be optimistic at best.

When you use **SAVE** and **LOAD**, the specifications and parameter values for the imputation equations will be restored with the data.

NOTE: No backwards incompatibilities with earlier versions of *LIMDEP* are created by this addition of material in the project file. The format of *LIMDEP* project files has never changed. You will be able to use **SAVE** and **LOAD** of project files across all versions of *LIMDEP* and *NLOGIT*.

R20.5 The Estimation Step

The imputation equations are used by placing the estimation step inside a procedure. You should first set the sample to be what you want to use for the estimation step irrespective of the missing values. Use **SAMPLE**, **REJECT**, **INCLUDE**, etc. to determine the current sample before doing the estimation. You will now define a procedure that includes the imputation as follows:

```
PROCEDURE $
Any commands that manipulate data, matrices, scalars, etc.
Model ; definition ; Imputation = a first imputation name … $
Any additional commands $
Model ; definition ; Imputation = a second imputation name … $
… repeated as many times as desired $
ENDPROCEDURE $
```

The procedure is then executed with

```
EXECUTE     ; N = desired number of imputations
            ; Imputation = first name, second name, … $
```

For a simple example,

```
PROCEDURE $
PROBIT      ; Lhs = doctor ; Rhs = one,age,educ,income ; Imputation = modela $
ENDPROCEDURE $
EXECUTE     ; N = 10 ; Imputation = modela $
```

The imputation identifiers act as follows: Imputation is used in generating the parameter vector and covariance matrix for that estimator. The procedure sets up the model and identifies it as one that will be using imputation. The procedure is unrestricted. It may contain any commands, including model commands. The **; Imputation = name** specification is used to identify those models that will be accumulating an average for the mean vector and covariance matrix. Thus, for example,

```
PROCEDURE $
PROBIT      ; Lhs = doctor ; Rhs = one,age,educ,income ; Imputation = modela $
POISSON     ; Lhs = hospital ; Rhs = one,age,healthy $
ENDPROCEDURE $
EXECUTE     ; N = 10 ; Imputation = modela $
```

would be a valid procedure. The Poisson regression does not use imputation, while the probit model does. (It would actually be a waste of effort to have the Poisson model inside the procedure, since it would be estimated the same 10 times, though only the last one would be reported.) Multiple imputation is used for models that produce an estimator and a covariance matrix. Commands that do not produce these may not include **; Imputation**. This would include commands such as **IDENTIFY**, **PLOT**, and **DSTAT**. **EXECUTE** may also include

```
; Report
```

to produce a line by line report of the progress of the estimation step. An example appears below.

There are no restrictions on what models may appear in the procedure. Every model in *LIMDEP* and *NLOGIT* is supported. The reason is that the imputation is created before the model command is carried out. The **EXECUTE** command, itself, fills in the missing values for each iteration, then any model that appears in the procedure can use the filled variables, as they are the names of variables that all exist in the data set. Within the procedure, the imputed data set becomes the new data set for all operations of the program.

Some care is required by this degree of flexibility. The imputed data set may still contain missing values if the imputation equation does not find complete data to fill all observations with gaps. The model commands, such as **POISSON** above, will be operating with a **SKIP** command on, so they will bypass remaining missing observations. However, data manipulations such as **MATRIX** will not automatically skip over missing data. You can use **REJECT** within the procedure, but it is a good idea to restore the sample to what it was at the beginning of the procedure if you do so.

The following shows an example of setting up a multiple imputation procedure. The first three commands just carry out an experiment. We randomly set about 30% of the values of variable income to -999 (missing).

```
SAMPLE      ; All $
CREATE      ; pick = Rnu(0,1) $
CREATE      ; If(pick < .3) income = -999 $
```

The next instruction sets up the imputation for this variable, using some other variables in the data set. The **; Type = measurement** indicates that the variable being filled is continuous, and the imputation equation is a linear regression.

```
IMPUTE      ; Lhs = income
            ; Rhs = one,age,educ,handper,married,working,bluec
            ; Type = measurement $
```

The brief summary of the imputation is the only output produced by the **IMPUTE** command.

```
----------------------------------------------------------
Equation stored for imputing missing values of     INCOME
Imputation method: Linear Regression
Observations currently in full data set        =    27326
Complete observations for imputation equation  =    19126
Missing observations on   HHNINC in data set   =     8200
----------------------------------------------------------
```

A more detailed summary of the imputation equation is obtained with the **List** instruction. The center column of the summary will indicate how many observations will have to remain unused because of missing values among the Rhs variables being used in the imputation equation. In this case, there are complete data for all of these variables (*age*, *educ*, *handper*, *married*, *working*, *bluec*).

IMPUTE ; List $

```
------------------------------------------------------------
Imputation Equations
Equations currently available for multiple imputation
The full sample contains       27326 observations
Complete observations counts apply to full sample
Missing obs for RHS is based on listwise deletion
Types: LR=linear regression, BR=binary logistic
       OL=ordered logit,     ML=multinomial logit
       FL=fractional logit,  PR=Poisson regression
---------------------+--------------+---------------------+
      Equation       |    RHS Vars  |Complete Observations|
LHS      type missing|Number Missing|   LHS    RHS     Eqn|
---------------------+--------------+---------------------+
HHNINC    LR     8200|   7         0| 19126  27326   19126|
---------------------+--------------+---------------------+
```

The procedure to do the estimation is as follows:

```
PROC = BnryChce $
LOGIT        ; Lhs = healthy ; Rhs = one,age,educ,income
             ; Imputation = impa $
PROBIT       ; Lhs = healthy ; Rhs = one,age,educ,income
             ; Imputation = impb $
ENDPROC $
EXECUTE      ; Proc = probit ; N = 5 ; Report
             ; Imputation = impa,impb $
```

Note that *income* appears in both models. The progress of the imputation based estimation appears after the **EXECUTE** command is submitted.

```
[Imputation] Begin executing   5 imputation/estimation procs.
[Imputation]  IMPA    ;   1 of   5. Estimated model with imputed data.
[Imputation]  IMPB    ;   1 of   5. Estimated model with imputed data.
[Imputation] Executed imputation/estimation procedure   2 of   5
[Imputation]  IMPA    ;   2 of   5. Estimated model with imputed data.
[Imputation]  IMPB    ;   2 of   5. Estimated model with imputed data.
[Imputation] Executed imputation/estimation procedure   3 of   5
[Imputation]  IMPA    ;   3 of   5. Estimated model with imputed data.
[Imputation]  IMPB    ;   3 of   5. Estimated model with imputed data.
[Imputation] Executed imputation/estimation procedure   4 of   5
[Imputation]  IMPA    ;   4 of   5. Estimated model with imputed data.
[Imputation]  IMPB    ;   4 of   5. Estimated model with imputed data.
[Imputation] Executed imputation/estimation procedure   5 of   5
[Imputation]  IMPA    ;   5 of   5. Estimated model with imputed data.
[Imputation]  IMPB    ;   5 of   5. Estimated model with imputed data.
[Imputation] Executed imputation/estimation procedure   6 of   5
Normal exit:   5 iterations. Status=0, F=    17398.41
 [Imputation] : Completed  5 estimations of IMPA     with imputed data.
--------------------------------------------------------------------------------
```

```
Binary Logit Model for Binary Choice
Dependent variable               HEALTHY
Log likelihood function    -17398.40698
Restricted log likelihood  -18279.94994
Chi squared [   3 d.f.]      1763.08592
Significance level               .00000
Estimation based on N =  27326, K =   4
MI results based on   5 imputed samples
Likelihood based stats are not reliable
when using multiple imputation methods.
Hosmer-Lemeshow chi-squared =  44.99363
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
 HEALTHY|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Characteristics in numerator of Prob[Y = 1]
Constant|     .91855***      .09430     9.74  .0000      .73373   1.10336
     AGE|    -.03901***      .00116   -33.53  .0000     -.04129   -.03673
    EDUC|     .10429***      .00610    17.08  .0000      .09233    .11626
  HHNINC|     .21566**       .10388     2.08  .0379      .01206    .41925
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

Normal exit:   4 iterations. Status=0, F=    17400.13
 [Imputation] : Completed  5 estimations of IMPB     with imputed data.

-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               HEALTHY
Log likelihood function    -17400.12943
Restricted log likelihood  -18279.94994
Chi squared [   3 d.f.]      1759.64102
Significance level               .00000
Estimation based on N =  27326, K =   4
MI results based on   5 imputed samples
Likelihood based stats are not reliable
when using multiple imputation methods.
Hosmer-Lemeshow chi-squared =  52.50448
P-value=  .00000 with deg.fr. =       8
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
 HEALTHY|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|     .58610***      .05684    10.31  .0000      .47470    .69750
     AGE|    -.02404***      .00071   -33.96  .0000     -.02543   -.02265
    EDUC|     .06243***      .00361    17.30  .0000      .05536    .06950
  HHNINC|     .13303**       .06344     2.10  .0360      .00870    .25737
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

[Imputation] Completed   5 imputation/estimation procs.

Maximum repetitions of PROC
```

R20.6 The Aggregation Step and Post Estimation Analysis

Final results are computed as follows for each model identified in the procedure as being estimated using imputed data: The parameter estimator of **β** is

$$\bar{\mathbf{b}} = \frac{1}{M}\sum_{m=1}^{M}\hat{\boldsymbol{\beta}}_m .$$

The variance estimator is

$$\bar{\mathbf{S}} = \frac{1}{M}\sum_{m=1}^{M}\hat{\boldsymbol{\Sigma}}_m + \left(1+\frac{1}{M}\right)\frac{1}{M-1}\sum_{m=1}^{M}\left(\hat{\boldsymbol{\beta}}_m - \bar{\mathbf{b}}\right)\left(\hat{\boldsymbol{\beta}}_m - \bar{\mathbf{b}}\right)'$$

where M is the number of imputations that are done. No other results are changed by the imputation. The estimation done at each replication treats the imputed data as if they were observed. In particular,

- No weighting scheme is used to reweight observations that contain imputed values.
- Account is taken of the possibility that the imputation might introduce a source of measurement error in the estimation.

The estimation results contain a warning that statistics such as likelihood ratio statistics might be unreliable when the parameter vector reported is an average of several coefficient vectors. However, you can use Wald (chi squared) tests as usual when you use multiple imputation. The tests can be included in the estimation command. For example,

```
PROBIT        ; Lhs = healthy ; Rhs = one,age,educ,income
              ; Test: age = 0 | age + income = 0
              ; imputation = impb $
```

would work as expected. The tests would be carried out based on the final (averaged) results after the imputation loop. You may also use the post estimation tools, **PARTIAL EFFECTS**, **SIMULATE** and **DECOMPOSE** within the imputation loop. The analyses will be delayed until after the imputation steps are completed. Thus, inference in each command is based on $\bar{\mathbf{S}}$. For example, the procedure could be

```
PROC = logitmdl $
LOGIT              ; Lhs = healthy ; Rhs = one,age,educ,income
                   ; Test: age = 0 | age + income = 0
                   ; Imputation = implogit $
PARTIAL EFFECTS    ; Effects: age & age = 20(5)65 $
ENDPROC $
EXECUTE            ; Proc = probit ; N = 5 ; Report ;  Imputation = implogit $
```

R20.7 Using Multiple Imputation in Your Own Model

You will generally produce your own models by using **MAXIMIZE/MINIMIZE**, **NLSQ**, **NLSURE**, **GMME**, etc. or by computing a parameter vector and covariance matrix using **MATRIX** with **CREATE**, **CALC**, and other programming commands. For the optimization commands such as **MAXIMIZE**, just include **; Imputation = name** in the command, the same as other model estimation commands. When you compute your own parameter vector and covariance matrix using **MATRIX**, use **DISPLAY** to show your results. For example, suppose your parameter vector and covariance matrix were named *beta* and *vb*. Then, after computing them, use

```
DISPLAY     ; Parameters = beta
            ; Covariance = vb
            ; Imputation = the name
            ; … any other options $
```

R20.8 Imputation Methods

The imputation methods used here build on Rubin's methods, with modifications for some of the types that he (and others) have not written about. In all cases, for each observation within each replication, we draw a random set of parameters from the posterior normal population. We then insert the prediction in place of the missing value. The original data set, with missing values, is restored after each iteration concludes. Each of the six estimators depends on an estimated set of parameters computed at the estimation step with an estimated asymptotic covariance matrix. Label these generically $\hat{\beta}$ and $\hat{\Gamma}$. Let the Cholesky decomposition of $\hat{\Gamma}$ be denoted **CC′**, so that **C** is a lower triangular matrix. With these in place, the simulations of the missing data are done as follows: For each case, a random draw on the parameter vector is generated by

$$\hat{\beta}_r = \hat{\beta} + \mathbf{C}\mathbf{w}_r$$

where $\mathbf{w}_r$ is a **K**-variate draw from the multivariate standard normal population. In each case, an applicable index function, $z_{i,r} = \hat{\beta}_r'\mathbf{z}_i$ is computed from the available data for that observation. Then, the following algorithms are used:

Binary Random Variable

Probabilities $P_0 = \Lambda(-z_{ir})$ and $P_1 = \Lambda(z_{ir}) = 1 - P_0$ are computed. The unit interval is divided into $[0,P_0]$ and $(P_0,1]$. A random draw from $U[0,1]$ is taken. If U is less than or equal to P_0 then $\hat{x}_{i,r} = 0$, otherwise $\hat{x}_{i,r} = 1$.

Fractional Random Variable

The random draw is simply $\hat{x}_{i,r} = \Lambda(z_{ir})$.

Continuous Random Variable

The random draw is

$$\hat{x}_{i,r} = z_{ir} + \hat{\sigma}\left(\sqrt{\frac{d.f.}{\chi^2_{i,r}}}\right)w_{i,r}$$

where $\chi^2_{i,r}$ is a random draw from the chi squared population with degrees of freedom equal to those that applied to the regression, and $w_{i,r}$ is a random draw from the standard normal population.

Ordered Outcome

The estimated ordered logit model includes threshold parameters $\hat{\mu}$. The estimated values are ordered, to insure all predicted probabilities are positive. The estimated values of $\hat{\mu}$ are used directly to maintain the coherence of the model. For the simulation, the J+1 cell probabilities

$$P_{j,ir} = \Lambda(\mu_j - z_{ir}) - \Lambda(\mu_{j-1} - z_{ir}),\ j = 0,1,\ldots,J \text{ are computed,}$$

where $\mu_0 = 0$ and $\mu_{-1} = -\infty$. The $J + 1$ cumulative probabilities are $Q_{j,ir} = \Lambda(\mu_j - z_{i,r})$. The unit interval is divided into the $J + 1$ cells by the values of $Q_{j,ir}$ where $Q_{-1,ir} = 0$. A random draw from $U[0,1]$ is obtained. If $U > Q_{j-1,ir}$ and $U \leq Q_{j,ir}$ then $\hat{x}_{i,r} = j$. For example, if $0 < U \leq \Lambda(-z_{ir})$ then $\hat{x}_{i,r} = 0$. If $\Lambda(-z_{ir}) < U < \Lambda(\hat{\mu}_1 - z_{ir})$ then $\hat{x}_{i,r} = 1$, and so on.

Poisson Distributed Count Variable

For a Poisson distributed outcome, we form the mean $\theta_{ir} = \exp(z_{ir})$. We then compute the CDF of the implied Poisson distribution $Q_j = Q_{j-1} + P(j)$ where $P(j)$ is the Poisson probability with mean θ_{ir} and $Q_0 = \exp(-\theta_{ir})$. A draw from the population is obtained by drawing a value from $U[0,1]$. If $Q_j \leq U < Q_{j+1}$, then $\hat{x}_{i,r} = j$.

Unordered Outcomes Multinomial Random Variable

The approach is essentially the same as that for the ordered outcome. For the multinomial logit model with J outcomes, the parameter vector is partitioned into J subvectors, $\boldsymbol{\beta}_j$ where $\boldsymbol{\beta}_J = \mathbf{0}$. Then, the J probabilities are computed according to

$$P_j = \frac{\exp(z_{ir,j})}{\Sigma_{j=1}^{J}\exp(z_{ir,j})}.$$

The [0,1] interval is partitioned into J parts with length $P_1, P_2,\ldots,P_J$. The intervals $[0,P_1)$, $[P_1,P_1+P_2)$ and so on are used to partition the unit interval into the J cells identified with $j = 1$, $j = 2$, and so on. A draw from $U[0,1]$ is then taken. If U falls in cell j, then $\hat{x}_{i,r} = j$.

R20.9 Usage Notes

The technique of multiple imputation is rooted in the Bayesian method of data augmentation. In this framework, missing values in an analysis are treated as unknown parameters to be estimated along with structural parameters. To consider a useful benchmark application, consider the probit model,

$$\begin{aligned} y^* &= \boldsymbol{\beta}'\mathbf{x} + \varepsilon \\ y &= 1[y^* > 0]. \end{aligned}$$

If y^* were observed directly, $\boldsymbol{\beta}$ would be estimated efficiently by least squares regression of y^* on $\mathbf{x}$. However, only y, not y^* is observed. In the classical treatment, under these circumstances, $\boldsymbol{\beta}$ is estimated by maximum likelihood. A Bayesian treatment (see Greene (2012, pp. 671-672)) can proceed by including y^* with $\boldsymbol{\beta}$ in the estimation process, using data augmentation. In general, the Gibbs sampler used in this method is based on estimating y^* by taking random draws from the distribution of y^* conditioned on the observed data and the current estimate of $\boldsymbol{\beta}$, then using regression of the predicted values of y^* on $\mathbf{x}$ to reestimate $\boldsymbol{\beta}$. The steps are repeated many times. This is essentially the strategy at work in multiple imputation. Estimation proceeds by estimating the missing values using estimates of the conditional model produced at the imputation step. The use of multiple imputations, as opposed to one single imputation, corresponds to the repetitions of the Gibbs sampler. The aggregation step, at which the multiple estimates of $\boldsymbol{\beta}$ are averaged, corresponds to the averaging of the draws on $\boldsymbol{\beta}$ by the Gibbs sampler.

R20.9.1 Questions on Usage

The theory of multiple imputation is well developed in great detail for the benchmark case of a linear regression of a variable y on a set of continuous variables $\mathbf{x}$, when the K+ 1 variables in the model are multivariate normally distributed. The theoretical foundation is less firm for the arguably more common cases in which the variables to be imputed are discrete, ordered, count variables, and so on, and the dependent variable in the model is likewise, discrete or otherwise not amenable to linear regression. Nonetheless, the underlying motivation is persuasive, and it does appear that some advantage can be gained through use of the technique when the data conditions are relatively favorable. We consider some general questions about MI here, but refer the reader to the applicable literature for technical details, more in depth analysis, and wisdom about usage.

Should I use multiple imputation instead of listwise deletion?

If there is any information about the missing values contained in the complete data, then listwise deletion obviously ignores this information. MI is likely to seem more attractive the greater is the number of incomplete observations. However, as the number of incomplete cases increases, the number of imputations necessary will increase accordingly. The danger of having the simulation noise taint the estimator increases as the relative presence of missing values increases.

Should I use multiple imputation instead of single imputation?

Single imputation amounts essentially to filling missing values either with a mean of the complete observations or with some sort of conditional mean such as an ancillary regression based on complete data. The case for MI instead of single imputation rests on the assumptions that motivate MI more generally. Estimation of standard errors based on a single imputation is likely to be more optimistic than appropriate. The estimated standard errors will not account for the presence of the sampling variability in the imputation equation.

How many imputations should I use?

This depends on many factors, including the model being estimated and the models used for the imputation. Under the most favorable case of multivariate normality and a relatively small number of incomplete cases, Rubin (1987) has argued that more than 90% of the advantage of MI can be obtained with $M = 2$. Since the most favorable case is also likely to be the least common, this provides only a floor. The literature has a focal point at $M = 5$ or 10. It is difficult to discern any useful rule of thumb for determining M. The suggestion that the analyst examine the behavior of their estimator with different values seems useful.

Should I use MI to impute missing values of the dependent variable?

Never.

Should I use complete values of the dependent variable in the imputations of the independent variables?

This is likely to invalidate the assumption that the variables on the right hand side of your equation are exogenous to the data generating process for the dependent variable. As a general proposition, the answer here would be no if you are interested in consistent estimation of the parameters of your model.

Is MI the same as Gibbs Sampling?

As described above, MI resembles Gibbs sampling. But, they are not the same thing. MI is not based on a Bayesian type of updating algorithm. The same imputation equation is used in every iteration of MI.

Is MI the same as the EM algorithm?

The EM algorithm is a method used for maximizing likelihood functions. It is firmly based on the distribution of the dependent variable and on the conditional distribution of the underlying unobserved data conditioned on the observed information. The probit example described in Section R20.1 bears resemblance to the EM method. The EM method also resembles the Gibbs sampler in the way that the weights constructed at the E (expectation) step are updated from one iteration to the next. Once again, since MI does not rely on any form of updating algorithm, for this reason and others, MI is not the same as the EM method.

R20.9.2 Implementation Notes

The MI feature of *LIMDEP* provides a looping procedure within which missing values of variables that you designate are filled with predictions from fitted models. Logically, internally, the program creates a map of the data area that records where missing values appear. As the iteration begins, the imputation equations are executed all at once to fill as many missing values as possible. Thus, a data set is prepared for the commands within the procedure to use. All commands within the procedure use the same imputed data set. At the end of the sequence of commands in the procedure, the data set is restored to its initial state, with missing values inserted where they were before. Each iteration of the procedure operates on a new imputed data set. A consequence of this procedure is that you do not have to do any data management of the imputed data at all. The output report will provide some information about how many missing values were filled and how many remained. But, absolutely no action is required of you to deal with the imputed data set.

You can achieve replicability of your imputation results by setting the seed of the random number generator before executing the procedure. You can also create imputed data sets by using the **IMPUTE ; Type = Fill… $** procedure described in Section R20.4 then using **EXPORT**, **SAVE** or **WRITE** to offload the data set.

We note a few computations this program does not do:

1. It does not examine and act on missing data ‘patterns,’ such as monotone missing values and so on. It fills the missing values in the variables, one variable at a time, independently. Given the way that the imputation step proceeds, patterns of missing data would not be useful. The program uses as many complete observations as it can find at every step. The knowledge that a variable *z*1 is missing whenever another variable, *z*2 is missing would not be useful.

2. It does not do any exotic corrections to degrees of freedom for the linear model. These will apply only to the linear model in any event, which is likely to be the least frequent case that one would use these methods. Moreover, the typical corrections for degree of missingness in the data usually produce values for the degrees of freedom parameter for the *t* distribution that are orders of magnitude beyond the point at which *t* becomes indistinguishable from standard normal. The inference framework in the MI setting is assumed to be reasonably approximated by asymptotic normality. Test statistics are based on large sample normal distributions and the chi squared distribution for Wald statistics. F ratios, used only in the linear model, are assumed to have denominator degrees of freedom large enough to use Wald statistics (J*F) instead.

3. We do not do any special data management, such as saving the imputed data set(s) as separate files. This is because we do not create freestanding replicated data sets with imputations. Imputations are done ‘on the fly,’ and fill in the gaps in the existing data set, in place. The advantage of this way of proceeding is that if you want to compute a thousand imputations with a huge data set, you can do it.

R21: Bootstrapping and Other Sampling Experiments

R21.1 Introduction

This chapter documents two functions available in *LIMDEP*, bootstrapping which will be useful for researchers and students interested in analyzing the properties of certain sample statistics, and sampling experiments, which will be primarily of interest to students and those using *LIMDEP* for instructional purposes.

R21.2 Bootstrapping Cross Sections and Panel Data

Bootstrapping is a technique used to deduce the properties (usually mean and variance) of the sampling distributions of estimators by using the variation in the observed sample under an assumption that the pattern of variation in the observed sample mimics reasonably accurately the counterpart in the population. The user is referred to one of the standard sources on this subject for discussion. Useful references include Efron (1979, 1998), Efron and Tibshirani (1986), Greene (2012) and other sources cited therein. The mechanics of the procedure are as follows: An estimator, **b** of a parameter or vector of parameters, $\boldsymbol{\beta}$ is computed using the data in a sample, $\mathbf{X} = [\mathbf{x}_1,...,\mathbf{x}_n]$. We desire to estimate the (usually asymptotic) sampling variance of the estimator, Asy.Var[**b**]. The technique is to compute

$$\mathbf{V}_{bs} = \frac{1}{R}\sum_{r=1}^{R}(\mathbf{b}_r - \mathbf{b})(\mathbf{b}_r - \mathbf{b})'$$

where R is the number of replications of the bootstrapping, and $\mathbf{b}_r$ is the estimate of $\boldsymbol{\beta}$ obtained from the rth bootstrap sample, using the same computations used to compute the original estimator, **b**. A bootstrap sample is obtained by sampling, *with replacement*, m observations from **X** (where m is generally, though not necessarily, equal to n). Note that the variation is computed around the original estimate, not the mean of the bootstrap estimates. Also, in what follows, we refrain from labeling $\mathbf{V}_{bs}$ the variance matrix, and instead call it the mean squared error. Under 'good' sampling conditions, $\mathbf{V}_{bs}$ is a reasonable estimator of Asy.Var[**b**].

The number of replications, R, needed depends on the quantity being estimated. For statistics with narrow precision, such as a specific quantile of the distribution, several hundred may be needed. For a broader characteristic, such as the asymptotic variance, research has found that perhaps 50 or 100 are likely to be sufficient.

Several of the preprogrammed estimators in *LIMDEP* use this technique to estimate the asymptotic variance of the estimator: **MSCORE**; **REGRESS** with the least absolute deviations estimator; **QREG** and **QCREG** which computes quantile regressions; the data envelopment analysis package in **FRONTIER**; and **MATCHING**, the propensity score matching estimator. You can use bootstrapping with *any* estimator that you compute with any model in *LIMDEP*, either with the preprogrammed commands, or one that you might program yourself using any of the programming tools.

The technique is used with the following steps:

Step 1. Write a procedure to compute the result. The result to be bootstrapped must either be a vector or a scalar. The procedure can involve any computation you wish, so long as the procedure computes the statistic you are analyzing and gives it a name.

Step 2. Execute the procedure as follows:

MATRIX or CALC	**;** initialize the entity being computed, to establish its name $
PROCEDURE	computes the parameter vector or scalar named above
EXECUTE	**; N** = number of bootstrap replications desired. (This is *R*.)
	; Bootstrap = the name of the statistic as declared above $

An optional specification in the **EXECUTE** command is

; Draws = number of observations in the bootstrapped sample.

This specifies *m*. If you omit this, the default value is the current sample size, *n*. If you do not specify the number of draws, the default is the original sample size. Note that the leading **MATRIX** or **CALC** is needed to establish what you are bootstrapping, as the output produced by this procedures differs in the two cases.

NOTE: When you use **; Bootstrap**, the **EXECUTE** command is carried out with **; Silent** set automatically. Only the original, full sample output is shown. You should not override this.

Replicating Bootstrapped Results

Bootstrapping is based on using the random number generators to produce pseudo-random samples. In order to be able to replicate an experiment, it is necessary to set the seed to the random number generator when the computations are begun. Use

; Seed = value

to do so. The value chosen is immaterial. All that is needed is to use the same value each time the procedure is executed.

Block Bootstrapping Panel Data

If your estimator is based on panel data, set up the panel with **SETPANEL** before any other operations. Then, you can include

; Panel or **; Pds = specification**

in the **EXECUTE** command. Rather than drawing individual observations in the bootstrapped sample, the procedure will sample groups of observations from the panel. The groups sizes may be different; the procedure works for balanced or unbalanced panels.

If the result your procedure computes is a vector, then the output of the procedure will be a statistical table consisting of the original estimate, estimates of the root mean squared errors (around the original estimate), *t* ratios and *p* values for the estimates. The latter two of these may be questionable, but will nonetheless be suggestive. If your result is a scalar, output will include a set of descriptive statistics, including the original estimate, root mean squared error, skewness, kurtosis, minimum, and maximum. You may also request a histogram of the bootstrapped values by adding

```
; Histogram
```

to the **EXECUTE** command.

Note, finally, you may use this set of procedures with any computation in any estimation program or programming tool. All that is required is that you

1. Declare the name of the vector or scalar before the **EXECUTE** command with a **MATRIX** or **CALCULATE** command.

2. Compute the vector or scalar, by name, within the procedure.

3. Bootstrap that specific vector or scalar with the **; Bootstrap** specification in the **EXECUTE** command.

Remaining output is handled for you by the program. How the computation is done within the procedure is completely up to you. There are no restrictions. All that is required is that the declared result be the output of a **MATRIX** or **CALC** command. For the first of these, that **MATRIX** command might do nothing more than simply capture an estimation result. For example:

```
NAMELIST    ; x = the variables on the Rhs of some model command $
CALC        ; k = Col(x) $
MATRIX      ; bb = Init(k,1,0.0) $     This is the declaration.
PROC $
  Model command to fit a model using x for which the coefficient vector is b $
MATRIX      ; bb = b  $                This captures the result.
ENDPROC $
EXEC        ; Bootstrap = bb ; N = 100 $
```

Two examples follow.

Bootstrapped Estimates of a Vector of Coefficients

LIMDEP computes standard errors for the marginal effects for its models by using the delta method. It has been argued that in small samples, the delta method may be a bit unreliable. In this example, we will examine this issue by doing the same computation using bootstrapping. The example is set up so that you should be able to replicate it exactly:

```
CALC        ; Ran(12347.0) $
SAMPLE      ; 1-50 $
CREATE      ; x1 = Rnn(0,1) ; x2 = Rnn(0,1) ; y = (x1 + x2 + Rnn(0,2)) > 0 $
NAMELIST    ; x = x1,x2,one $
MATRIX      ; mrgfct = Init(2,1,0) $          Note that this declares vector mrgfct.
PROC  $
PROBIT      ; Lhs = y ; Rhs = x ; Partial Effects $
CREATE      ; scale = N01(b'x) $
MATRIX      ; mrgfct = {Xbr(scale)} * b(1:2) $  This computes vector mrgfct.
ENDPROC $
EXECUTE     ; Bootstrap = mrgfct ; N = 100 $    This bootstraps mrgfct.
```

The **CREATE** and **MATRIX** commands after the **PROBIT** command obtain the average partial effects for the two variables. This experiment produces the following output: In fact, the standard errors computed by the delta method are quite close to the bootstrapped values. The estimated marginal effects and estimated standard errors computed by the probit estimator based on the full sample are reported after the results of the bootstrap procedure. Note that the confidence interval reported for the bootstrap results is almost the same as that for the probit model which used the delta method. (The effects themselves are identical because the bootstrap procedure reports the original coefficients based on the actual sample. Only the standard errors are bootstrapped.)

```
Completed    100 bootstrap iterations.
-----------------------------------------------------------------------------
Results of bootstrap estimation of model.
Model has been reestimated   100 times.
Coefficients shown below are the original
estimates based on the full sample.
bootstrap samples have   50 observations.
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
BootStrp|  Coefficient      Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
MRGFC001|     .18647***      .05117     3.64  .0003      .08618    .28676
MRGFC002|     .06452         .05432     1.19  .2350     -.04195    .17099
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable                    Y
Log likelihood function       -29.76776
Restricted log likelihood     -34.01460
Chi squared [   2 d.f.]         8.49369
Significance level               .01431

--------+--------------------------------------------------------------------
        |                                      Prob.      95% Confidence
       Y|  Coefficient   Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
      X1|     .55286***      .00209     2.63  .0085      .14137    .96434
      X2|     .19129         .25802     1.08  .2803     -.15598    .53856
Constant|     .18840        1.00000      .98  .3277     -.18887    .56567
--------+--------------------------------------------------------------------
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with
respect to the vector of characteristics
Average partial effects for sample obs.
--------+--------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
       Y|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
      X1|     .18647***      .00067     3.44  .0006      .08032    .29262
      X2|     .06452         .02861     1.12  .2634     -.04855    .17758
--------+--------------------------------------------------------------------
#  Partial effect for dummy variable is E[y|x,d=1] - E[y|x,d=0]
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Notice that the labels for the bootstrap coefficients in the output table are constructed from the name given in the **EXECUTE** command. In the example, **; Bootstrap = mrgfc** and the two coefficients are labeled *mrgfc*001 and *mrgfc*002. You can provide a specific set of labels to be used in the results with

; Labels = the list of labels.

This list may be a labellist defined earlier with a **CLIST** command. If the labels correspond to a set of variables, as in a regression style model, you could also give the namelist as the labellist, and the variable names will appear in the output.

The Distribution of a Sample Correlation Coefficient

Applying the delta method to the distribution of a sample correlation coefficient as a function of the sample variances and covariance is notoriously unreliable. A standard approximation, Asy.Var[r] = $(1 - r^2)/n$, is known to be problematic in small samples and when ρ is close to 1.0. The following analyzes the correlation between two variables empirically.

The commands are:

```
CALC          ; Ran(12347.0) $
SAMPLE        ; 1-500 $
CREATE        ; x1 = Rnn(0,1) ; x2 = x1 + Rnn(0,1)  $
CALC          ; r12 = 0 $
PROC  $
CALC          ; r12 = Cor(x1,x2)  $
ENDPROC $
EXECUTE       ; Bootstrap = r12 ; N = 500 ; Histogram $
```

Results are shown below. The true correlation is 1/√2 ≈ 0.702. The sample estimate is 0.720. In fact, the distribution does look noticeably nonnormal. The standard approximation, Asy.Var[r] = (1 – r^2)/n would produce an approximation of 0.031 for the asymptotic standard error. The bootstrapped root mean squared deviation is 0.020, which is quite different.

```
Completed   500 bootstrap iterations.
+-------------------------------------------+
| Results of bootstrap estimation of model.|
| Model has been reestimated   500 times.  |
| Statistics shown below are centered      |
| around the  original estimate  based on  |
| the original full sample of observations.|
| Result is R12      =       .71951        |
| bootstrap samples have  500 observations.|
| Estimate  RtMnSqDev  Skewness   Kurtosis |
|     .720       .020     -.314      3.266 |
| Minimum =      .635  Maximum =      .777 |
+-------------------------------------------+
```

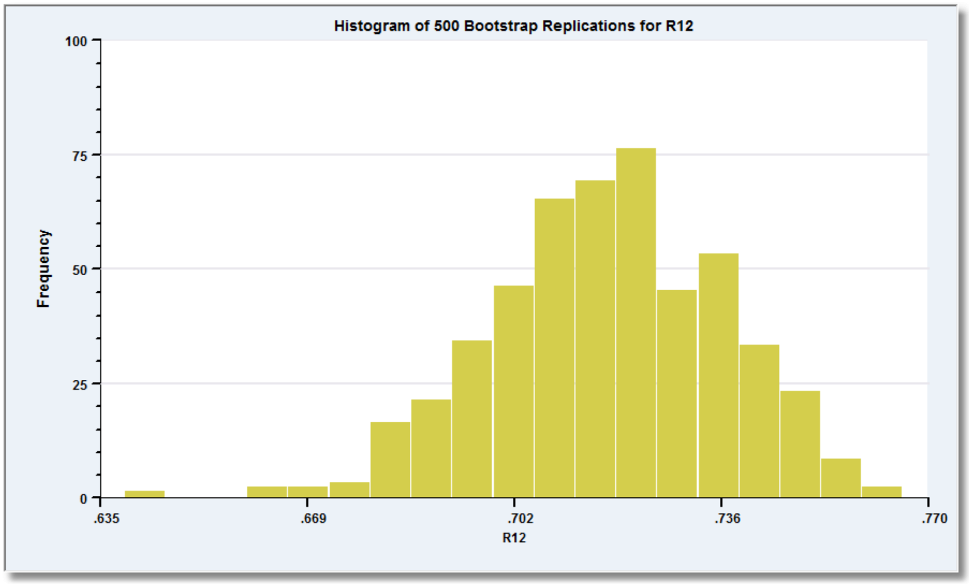


Figure R21.1 Estimated Distribution of the Sample Correlation Coefficient

R21.3 Jackknife Estimation

This procedure computes the variance (matrix) for an estimator by jackknife replications. In a sample of *N* observations, the estimator is computed *N* times, dropping one observation from the sample each time. The variance of the resulting estimators is reported with the original estimator. The jackknife estimator of the variance of the estimator is

$$\mathbf{V}_{bs} = \frac{N}{N-1}\sum_{i=1}^{N}\left(\mathbf{b}_{(i)} - \mathbf{b}\right)\left(\mathbf{b}_{(i)} - \mathbf{b}\right)'$$

where **b** is the full sample estimator of β and $\mathbf{b}_{(i)}$ denotes the estimator computed using all observations except for observation *i*.

Jackknife estimation may be used with any estimator that you compute using a sample of data – that includes all model estimators or any other statistic that is computed with a sample of observations – it operates the same as the bootstrap estimator described above. Any specifications, options for the estimator (such as clustering, robust variance matrices), etc. may be used. The jackknife estimator replications are computed using the precisely identical setup as the original.

The specification **; Jackknife = entity** is used for this technique. The template procedure for using the jackknife estimator is:

```
PROCEDURE $
        any model or other computation that computes the estimator or statistic,
        and gives it a name $
ENDPROCEDURE $
EXECUTE     ; Jackknife = name $
```

Note, again, this is otherwise the same as the setup for the bootstrap estimator.

A small example that computes a variance estimator for the least squares coefficient follows. A second example computes a jackknife estimator for the mean of a sample of functions of observations drawn from a chi squared distribution.

To illustrate the jackknife estimator, we will estimate the coefficients of a probit model using a widely used sample of 32 observations on performance by high school students. (See Greene (2012, p. 590). Variables *gpa*, *tuce*, *psi* and *grade* are grade point average, a test score, a treatment dummy for a type of learning program, and whether grades in an Economics course improved.

OBS	GPA	TUCE	PSI	GRADE	OBS	GPA	TUCE	PSI	GRADE
1	2.66	20	0	0	2	2.89	22	0	0
3	3.28	24	0	0	4	2.92	12	0	0
5	4.00	21	0	1	6	2.86	17	0	0
7	2.76	17	0	0	8	2.87	21	0	0
9	3.03	25	0	0	10	3.92	29	0	1
11	2.63	20	0	0	12	3.32	23	0	0
13	3.57	23	0	0	14	3.26	25	0	1
15	3.53	26	0	0	16	2.74	19	0	0
17	2.75	25	0	0	18	2.83	19	0	0
19	3.12	23	1	0	20	3.16	25	1	1
21	2.06	22	1	0	22	3.62	28	1	1
23	2.89	14	1	0	24	3.51	26	1	0
25	3.54	24	1	1	26	2.83	27	1	1
27	3.39	17	1	1	28	2.67	24	1	0
29	3.65	21	1	1	30	4.00	23	1	1
31	3.10	21	1	0	32	2.39	19	1	1

The jackknife estimator is shown below.

```
PROBIT     ; Lhs = grade ; Rhs = one,gpa,tuce,psi $
PROC $
PROBIT     ; Lhs = grade ; Rhs = one,gpa,tuce,psi ; Quiet $
ENDPROC $
EXECUTE    ; Jackknife = b $
```

```
-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable                GRADE
Log likelihood function       -12.81880
Restricted log likelihood     -20.59173
Chi squared [   3 d.f.]        15.54585
Significance level               .00140
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
   GRADE|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|   -7.45232***     2.54247    -2.93  .0034   -12.43547   -2.46917
     GPA|    1.62581**       .69388     2.34  .0191      .26583    2.98579
    TUCE|     .05173         .08389      .62  .5375     -.11269     .21615
     PSI|    1.42633**       .59504     2.40  .0165      .26008    2.59259
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

Completed    32 jackknife iterations.

-----------------------------------------------------------------------------
Results of jackknife estimation of model.
Model has been reestimated    32 times.
Coefficients shown below are the original
estimates based on the full sample.
jackknife samples have   31 observations.
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
JckKnife|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
    B001|   -7.45232        4.76289    -1.56  .1177   -16.78741    1.88277
    B002|    1.62581        1.12779     1.44  .1494     -.58463    3.83625
    B003|     .05173         .10033      .52  .6061     -.14491     .24837
    B004|    1.42633*        .73717     1.93  .0530     -.01850    2.87117
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

This example considers estimation of $E[f(x)]$ where x has a chi squared distribution with two degrees freedom and $f(x) = .5*\ln x / \exp(.2x^2)$. We draw a sample of 100 observations from the chi squared population, compute the functions, the mean of the functions, and the bootstrap estimator of the variance of the statistic.

For comparison, the conventional estimator, $N^{-1/2} \times$ Std.Dev(f(x)) is reported as well.

```
CALC        ; Ran(12359) $
SAMPLE      ; 1-100 $
CREATE      ; x = Rnx(2) $
CREATE      ; x = Rnx(2) $
CREATE      ; f_x = .5*Log(x)/Exp(.2*x^2) $
PROC $
CALC        ; meanf_x = Xbr(f_x) $
ENDPROC $
EXECUTE     ; Jackknife = meanf_x $
```

```
Completed   100 jackknife iterations.
+--------------------------------------------+
| Results of jackknife estimation of model.|
| Model has been reestimated   100 times.  |
| Statistics shown below are centered      |
| around the  original estimate  based on  |
| the original full sample of observations.|
| Result is MEANF_X  =      -.14928        |
| jackknife samples have   99 observations.|
| Estimate  RtMnSqDev  Skewness   Kurtosis |
|    -.149       .049      .003       .001 |
| Minimum =     -.152  Maximum =     -.124 |
+--------------------------------------------+
Maximum repetitions of PROC
```

```
CALC        ; List ; samplesd = 1/Sqr(n) * Sdv(f_x) $
```

```
[CALC] SAMPLESD=       .0483802
```

The bootstrap value of .049 compares to the empirical estimate of .0483. As noted, this estimator can be employed with any statistic or estimator that you compute with a sample. One exception: This estimator should not be used with panel data estimators. The ‘leave one out’ procedure leaves out an observation, not a group of observations. This is useable for cross sections, but not for panels. Its validity in time series is ambiguous, since the resampling procedure ‘punches a hole’ in the time series.

A ‘bias correction’ has been suggested for the jackknife estimator. The corrected estimator is

$$\mathbf{b}_c = \mathbf{b}_0 + \frac{1}{n}\mathbf{b}_0 - \frac{1}{n-1}\overline{\mathbf{b}}$$

Where b_0 is the original estimator and $\overline{\mathbf{b}}$ is the mean of the jackknife replicates. Request this correction with

```
; bias correction
```

R21.4 Random Sampling from the Current Sample – DRAW

One of the important components of the bootstrapping procedure described in the previous section is the process of sampling observations from the current 'master' sample. The bootstrap feature of **EXECUTE** does this automatically. You may wish to draw a random sample from your data set for other reasons, for example, to program a different type of bootstrap estimator of your own. You can draw a random sample *from the current sample of observations* with the **DRAW** command. The procedure is as follows: First, set the parent population to whatever is desired with **READ**, **SAMPLE**, **REJECT**, and **INCLUDE**. This results in *nobs* observations. The command to draw a random sample is

```
DRAW        ; N = number $
```

to sample *number* observations without replacement. **N** must be less than *nobs*. Use

```
DRAW        ; N = number ; Rep $
```

to sample with replacement. In this case, *nobs* can be anything and *number* can be up to 100,000. For example:

```
SAMPLE      ; 1-100 $
CREATE      ; i = Trn(1,1) $      numbers from 1 to 100 will display
LIST        ; i $                 numbers from 1 to 100 in order
DRAW        ; N = 200 ; Rep $
LIST        ; i $                 will display 200 random draws from i
```

The original data are not changed, only the sample pointers are. Restore the original sample with

```
DRAW        ; N = 0 $
```

All commands which modify the sample turn off the random sample and restore the original data set. These are **REJECT**, **INCLUDE**, **SAMPLE**, **DATES**, **PERIOD**.

WARNING: Do not do any operation which modifies your existing data while this sampling procedure is in effect. The results will be unpredictable and can be severely problematic. This affects *all* operations that use the data.

WARNING: Do not use **SKIP** with bootstrapped samples or random samples. **SKIP** generates an internal **REJECT** command which will then automatically produce a **DRAW ; N = 0 $** command even if no observations get skipped.

You can also enter a **DRAW** random sample command by choosing Project:Set Sample

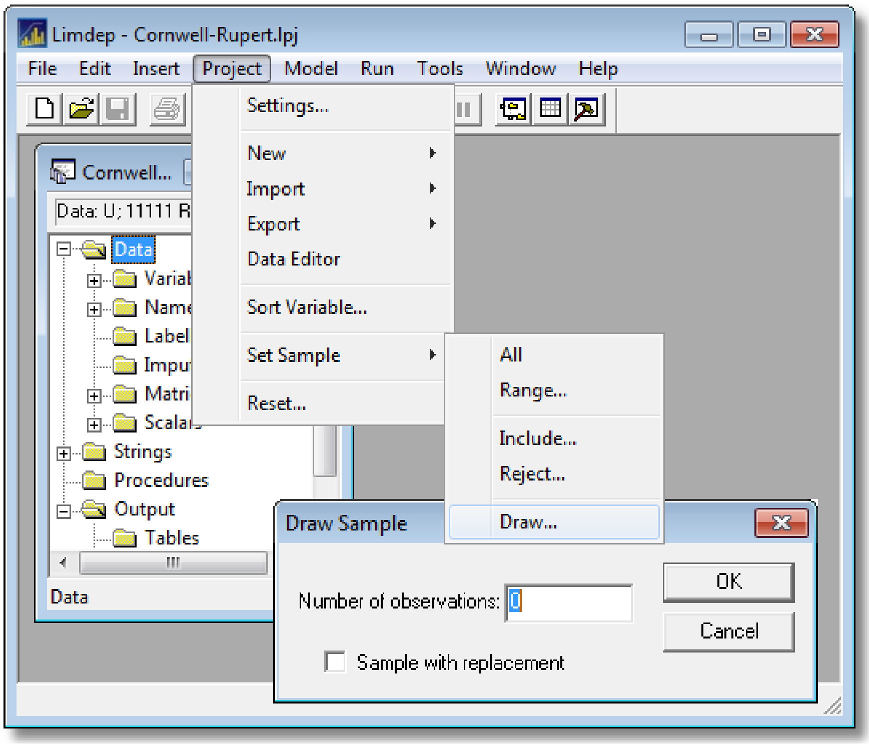


Figure R21.2 Dialog Box for DRAW Command

R21.5 Random Sampling from Panel Data Sets

You can draw a sample from a panel data set as easily as a cross section, by using **DRAW** as described above. However, this will probably not produce the effect you want. In a sample of data $\{\mathbf{z}_{i,t},\ t = 1,...,T_i,\ i = 1,...,N\}$, one will typically want to sample over i, not over i and t. The random sample will consist of N_s sampled individuals, with group sizes T_i equal to the original group sizes. For a concrete example, if you have a sample of 1,000 firms, and five observations on each firm, you will generally want to sample N_s firms, then for each firm sampled, the draw consists of all five observations for this firm. Simply using **DRAW** as defined in the preceding section will interrupt this sample configuration. *LIMDEP*'s **DRAW** command provides an option to allow this sort of sampling. To use this, if you have an unbalanced panel, you must have a group count variable available, as discussed in Chapter R5. For balanced panels, you need only provide the fixed group size. In both cases, the command is

```
DRAW        ; N = number of groups to sample
            ; Pds  =  the panel specification
            [ ; Replacement ] $
```

The last specification is optional. If the panel has a fixed group size, then **; Pds = the group size**, for example, **; Pds = 5**. If the panel is unbalanced, then **; Pds = the name of the group size variable**. There are internal limits on the size of panel that may be sampled:

- The overall, total sample size must be $\leq$ 750,000.
- The number of groups sampled must be $\leq$ 20,000.

R21.6 Random Number Generators

The other crucial component of bootstrapping and **DRAW**, as well as any other sampling experimentation you might do, is the set of random number generators. You can generate random numbers from a variety of distributions and with all of **CREATE** (columns of values in a variable), **CALC** (single random draws) and **MATRIX** (matrices of random values).

R21.6.1 Setting the Seed for the Random Number Generator

You can produce replicability for any computation involving random number generation by using a particular value of the seed for the generator. Note, for example, that the two examples above begin by setting the seed to a particular value; this way, you can reproduce our results.
To reset the seed for the random number generator, use the command

```
CALC        ; Ran(seed) $
```

In this fashion, you can replicate a sample from one session to the next. Use a large odd number for your seed.

R21.6.2 Using CREATE to Generate Random Samples

There are numerous transformations which draw samples using *LIMDEP*'s random number generators. The basic generators are for the continuous uniform distribution in the indicated range and the normal distribution with a specific mean and standard deviation. The commands are

```
      CREATE      ; name = Rnu(lower limit, upper limit) $

and   CREATE      ; name = Rnn(mean, standard deviation) $
```

which will create a variable containing a sample from the indicated normal distribution. The central tool for discrete distributions is the discrete uniform generator,

```
CREATE      ; name = Rnd(upper limit) $
```

which draws values randomly from the set 1, 2, ..., upper limit. Thus, for example, to simulate coin tosses with heads = 1 and tails = 2, you would use Rnd(2).
The sample is placed with the observations in the current sample. If you want to draw more than the default number, you might want to use the **ROWS** command (See Section R3.4) before you draw the sample.
Random draws may also appear anywhere in an expression as operands whose values are random draws from the specified distribution. For example, a random sample from a chi squared distribution with one degree of freedom could be drawn with

```
CREATE      ; name = Rnn(0,1) ^ 2 $
```

Random samples can be made part of any other transformation. For example, the following shows how to create a random sample from a regression model in which the assumptions of the classical model are met exactly:

```
CREATE      ; x1 = Rnu(10,10)
            ; x2 = Rnn(16,10)
            ; y  = 100 + 1.5 * x1 + 3.1 * x2 + Rnn(0,50) $
```

The regression of *y* on *x*1 and *x*2 would produce estimates of $\beta_1 = 100$, $\beta_2 = 1.5$, and $\beta_3 = 3.1$.

In addition to the Rnn(*m*,*s*) (normal with mean *m* and standard deviation *s*) and Rnu(*l*,*u*) (continuous uniform between *l* and *u*), you may use these additional generators in the same fashion to sample from the normal family of distributions:

Rng(*m*,*s*)	= lognormal with parameters *m* and *s*,
Rnt(*d*)	= *t* with *d* degrees of freedom,
Rnx(*d*)	= chi squared with *d* degrees of freedom.

Use

```
CREATE      ; F = (Rnx(dn)/dn)/(Rnx(dd)/dd) $
```

for the F distribution with *dn* and *dd* degrees of freedom.

In addition, you can use

Rne(*q*)	= exponential with mean *q*,
Rnw(0)	= Weibull,
Rnl(0)	= logistic,
Rnc(0)	= Cauchy,
Rnp(*q*)	= Poisson with mean *q*,
Rnd(*n*)	= discrete uniform, *x* = 1,...,*n*,
Rnb(*n*,*p*)	= binomial, *n* trials, probability *p*,
Rnh(*a*,*c*)	= Gumbel (extreme value) with location *a*, scale *c*. If *c* = 1, use Rnh(*a*).
Rni(*a*,*c*)	= gamma with scale *a* and shape *c*. If *a* = 1, use Rni(*c*).
Rna(*a*,*b*)	= beta with parameters *a* and *b*.

You must provide the ‘0’ in the Weibull, logistic, and Cauchy functions. You may also sample from the truncated standard normal distribution. Two formats are

Rnr(*lower*)	= sample from the distribution truncated to the left at ‘lower,’
Rnr(*lower*,*upper*)	= distribution with both tails truncated.

E.g., Rnr(.5) samples observations greater than or equal to .5.

Parameters of all requests for random numbers are checked for validity. For the truncated normal, you must have

$$lower \leq 1.5,\ upper \geq -1.5,\ upper - lower \geq .5.$$

If *upper* is not provided, it is taken as $+\infty$. If you need upper truncation, a transformation which will produce the desired result is-Rnr(-*lower*).

The parameters of any random number generator can be variables, other functions, or expressions, as well. For example, you might simulate draws from a Poisson regression model with

```
CREATE        ; x1 = Rnn(0,1)
              ; x2 = Rnu(0,1)
              ; y  = Rnp(Exp(.2 + .3 * x1 - .05 * x2 )) $
```

R21.6.3 Sampling from the Multivariate Normal Distribution

To sample from the multivariate normal distribution, it is necessary to generate a set of random variables. We do this by using the following theoretical result.

> If $\mathbf{x} = (x_1,...,x_K)$ are distributed with joint normal distribution with mean vector $\mathbf{0}$ and covariance matrix $\mathbf{I}$, then $\mathbf{Ax} + \boldsymbol{\mu}$ is distributed multivariate normally with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{AA}'$.

You can use this result to generate a multivariate sample from the normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ by simply decomposing $\boldsymbol{\Sigma}$ into $\mathbf{AA}'$, and using this and the desired $\boldsymbol{\mu}$ in the theoretical result. We use the Cholesky decomposition in which $\mathbf{A}$ is a lower triangular matrix. The operation will create a multivariate sample – that is K variables where K is the number of elements in $\mathbf{x}$ and N observations, where N is the number of observations in the current sample. You can sample from the distribution with up to 100 elements, in which case, you will create 100 new variables in your data area. Collectively, these K variables are a multivariate sample from the specified multivariate normal distribution.

The command for generating a sample from the multivariate normal distribution is

```
CREATE        ; name = Rmn(vector μ, matrix Σ) $
```

You must provide the vector$\boldsymbol{\mu}$ and matrix $\boldsymbol{\Sigma}$. However, if you want $\boldsymbol{\mu}$ to equal zero, omit it. Thus,

```
CREATE        ; name = Rmn(matrix Σ) $
```

samples from the multivariate normal population with mean vector zero and covariance matrix $\boldsymbol{\Sigma}$. Alternatively, you can force $\boldsymbol{\Sigma}$ to be an identity matrix by using

```
CREATE        ; name = Rmn(vector μ) $
```

to sample from the multivariate normal population with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{I}$. Finally, if you want to sample from the standard normal population with mean vector zero and covariance matrix $\mathbf{I}$, use

```
CREATE        ; name = Rmn(k) $
```

where k is the number of elements in the random vector. In this case, k must either be an integer from 1 to 100 or the name of scalar which contains an integer from 1 to 100. *LIMDEP* detects what kind of sample you want to generate by examining what appears in the parentheses. A vector and a matrix implies the first case, just a matrix implies the second, just a vector, the third, and just a number, the fourth.

The '**; name =**'specifies the name of a namelist that will be created. This may be a new namelist or you can replace an existing one. The variables in that namelist will be constructed as if the command were

NAMELIST ; name = name00, name01,... $

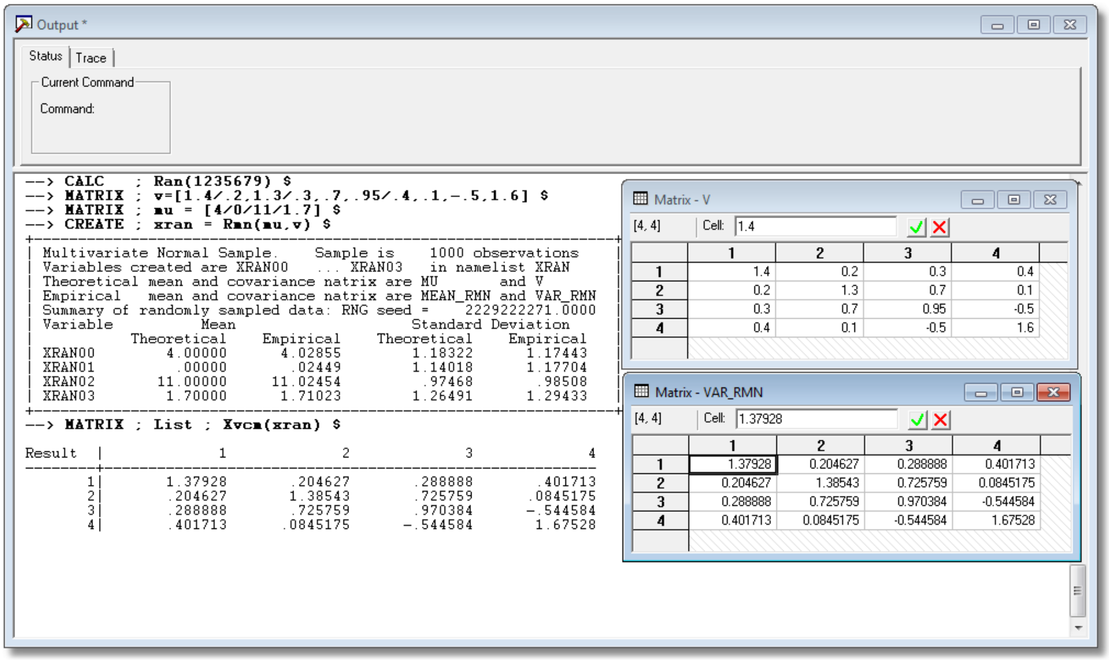
For example, if you use

CREATE ; xret = Rmn(mu,v) $

where *mu* is a 10×1 vector and *v* is a 10×10 covariance matrix, then there will be a new namelist created in your data area:

xret = *xret*00,*xret*01,*xret*02,...,*xret*09.

This routine creates the variables, and issues a report of what it has computed. The following shows an example of sampling 1,000 observations is from a 4-variate normal distribution.



```
--> CALC   ; Ran(1235679) $
--> MATRIX ; v=[1.4/.2,1.3/.3,.7,.95/.4,.1,-.5,1.6] $
--> MATRIX ; mu = [4/0/11/1.7] $
--> CREATE ; xran = Rmn(mu,v) $
+--------------------------------------------------------------------------+
| Multivariate Normal Sample.    Sample is    1000 observations            |
| Variables created are XRAN00   ... XRAN03   in namelist XRAN             |
| Theoretical mean and covariance natrix are MU       and V                |
| Empirical   mean and covariance natrix are MEAN_RMN and VAR_RMN          |
| Summary of randomly sampled data: RNG seed =    2229222271.0000          |
| Variable          Mean                      Standard Deviation           |
|             Theoretical    Empirical    Theoretical    Empirical         |
| XRAN00          4.00000      4.02855        1.18322      1.17443         |
| XRAN01           .00000       .02449        1.14018      1.17704         |
| XRAN02         11.00000     11.02454         .97468       .98508         |
| XRAN03          1.70000      1.71023        1.26491      1.29433         |
+--------------------------------------------------------------------------+
--> MATRIX ; List ; Xvcm(xran) $

Result  |          1            2            3            4
--------+--------------------------------------------------
       1|    1.37928      .204627      .288888      .401713
       2|    .204627      1.38543      .725759     .0845175
       3|    .288888      .725759      .970384     -.544584
       4|    .401713     .0845175     -.544584      1.67528
```

Matrix - V

[4, 4] Cell: 1.4

	1	2	3	4
1	1.4	0.2	0.3	0.4
2	0.2	1.3	0.7	0.1
3	0.3	0.7	0.95	-0.5
4	0.4	0.1	-0.5	1.6

Matrix - VAR_RMN

[4, 4] Cell: 1.37928

	1	2	3	4
1	1.37928	0.204627	0.288888	0.401713
2	0.204627	1.38543	0.725759	0.0845175
3	0.288888	0.725759	0.970384	-0.544584
4	0.401713	0.0845175	-0.544584	1.67528

Figure R21.3 Sampling from the Multivariate Normal Population

Note in the report in the output window, the theoretical and empirical means and variances are both reported. The actual mean and standard deviations of the drawn sample will not equal the theoretical ones, since the data are a random sample – they are not constrained. Also, the report shows the seed for the random number generator. It does not equal the seed that appears in the command in the editing window. The **CALC ; Ran(seed) $** function allows you to set a specific seed for the random number generator. The actual value used internally is a transformation of the one you give. The point of the function is to enable you to reset the seed to the same value, not a particular value. Specific values of the seed are meaningless. But, your ability to reset the seed to a specific value allows you to replicate random sampling results.

This procedure creates several results:

- The namelist as specified in the command,
- The variables (up to 100 of them) which are the random sample,
- Matrices *mean_rmn* which is the matrix of means of your sample, and *var_rmn* which is the sample covariance matrix.

The latter two matrices could be created immediately after the sampling command with

```
MATRIX       ; mean_rmn = Mean(namelist)
             ; var_rmn = Xvcm(namelist) $
```

All of the elements of the setup for this computation are checked internally before any computation is done. The following conditions will generate diagnostics:

- Your matrices *mu* and *v* are not currently in the matrix names table.
- Your parentheses contain more than two names.
- The matrix is not square.
- The vector is not conformable with the matrix; *mu* may be a row or a column, but it must be the same size as *v* whichever applies.
- Your computation implies more than 100 variables.
- You are out of space for new namelists or variables.
- Your matrix *v* is not symmetric.
- Your matrix *v* is not positive definite.

If none of these failures occur, the computation will proceed.

R21.6.4 Using CALC to Generate Random Draws

CALC has a limited facility for random sampling. The Rnn, Rnu, and Rnd functions as described for **CREATE** are also available for **CALC**. Thus,

```
CALC         ; name = Rnu(lower limit, upper limit) $
CALC         ; name = Rnn(mean, standard deviation) $
CALC         ; name = Rnd(upper limit) $
```

compute single draws from the continuous uniform, normal, and discrete uniform distributions, respectively. As in **CREATE**, these may be transformed to sample from other distributions.

R21.6.5 Using MATRIX to Draw Random Matrices

The **MATRIX** command **; a = Rndm(list)** can be used to draw matrices of random numbers from the normal distribution. The following specifications may be used:

Rndm(*m*)	= *m*×1 random vector from standard normal,
Rndm(*r*,*m*)	= *r*×*m* random matrix from standard normal.

All elements are independent draws. You may also specify the mean vector and covariance matrix for a draw of a random vector from the normal distribution:

Rndm(*mu*)	= *r*×1 random vector from normal distribution with mean *mu* and covariance matrix **I**. The matrix *mu* may be a row or column vector, and *r* is the number of elements in *mu*.
Rndm(*sigma*)	= *r*×1 random vector from multivariate normal distribution with mean vector **0** and covariance matrix *sigma*. The number of rows in *sigma* is *r*. You must provide a positive definite *sigma* matrix.
Rndm(*mu*,*sigma*)	= *r*×1 random vector from multivariate normal distribution with mean vector *mu* and covariance matrix *sigma*. The matrix *mu* must be the name of a row or column vector with *r* elements, and *sigma* must be the name of a square matrix with *r* rows.

R21.6.6 Simulating Random Effects in a Panel

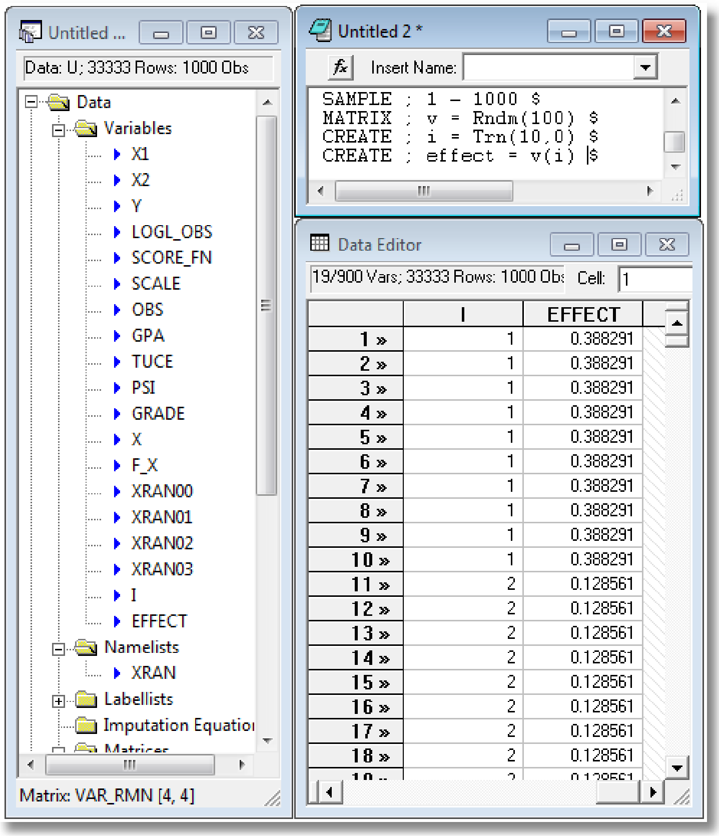
The random number generator in **MATRIX** can be used to simulate random effects models. Suppose your panel (real or simulated) has *n* groups, and each group has *T*(*i*) individuals. You will require a stratification indicator that is incremented from one to do this. You can easily obtain this if you do not already have it with

```
REGRESS     ; Lhs = one ; Rhs = one ; Pds = group count  or
            ; Str  = the stratum indicator you have
            ; Panel $
```

This creates *_stratum* which will be the variable you need. Now, suppose you know the theoretical standard deviation of the common effect, say *sd*. Then, you can use

```
MATRIX      ; v_i = Rndm(your n) $
CREATE      ; ui   = sd * v_i ( _stratum) $
```

The example in Figure R21.4 computes the random effect for a balanced panel with 10 observations per individual.



```
SAMPLE ; 1 - 1000 $
MATRIX ; v = Rndm(100) $
CREATE ; i = Trn(10,0) $
CREATE ; effect = v(i) $
```

	I	EFFECT
1 »	1	0.388291
2 »	1	0.388291
3 »	1	0.388291
4 »	1	0.388291
5 »	1	0.388291
6 »	1	0.388291
7 »	1	0.388291
8 »	1	0.388291
9 »	1	0.388291
10 »	1	0.388291
11 »	2	0.128561
12 »	2	0.128561
13 »	2	0.128561
14 »	2	0.128561
15 »	2	0.128561
16 »	2	0.128561
17 »	2	0.128561
18 »	2	0.128561

Figure R21.4 Simulating a Random Effect

R21.6.7 Simulating an Unbalanced Panel Data Set

If you are generating random samples using the random number generators, you will want to be able to simulate the group sizes for the simulation. This is a fairly involved operation. The following procedure will do it (for a balanced panel as well.)

```
CALC       ; ni = ... the number of groups you want in your panel $
SAMPLE     ; 1 - ni $
CREATE     ; ti = Rnd(m) $  Set m to the largest group size you want.
MATRIX     ; mti = ti $
CALC       ; i1 = 1 ; i = 1 ; sumti = 0 $
PROC $
CALC       ; i2 = i1 + mti(i) - 1 $
SAMPLE     ; i1 - i2 $
CREATE     ; ... < the variables you want to simulate> ...
             ...  $
CREATE     ; groupti = mti(i) ; groupid = i $
CALC       ; sumti = sumti + mti(i) ; i1 = i1 + 1 ; i = i + 1 $
ENDPROC $
EXECUTE    ; N = ni $
SAMPLE     ; 1 - sumti $
```

You can now analyze these panel data. Use **; Pds = groupti** for group size counts. *Groupid* is a simple (1,2,...) group identifier. For example, you could use *groupid* in the calculation of individual random effects described in the preceding section.

R21.7 Plotting Distributions

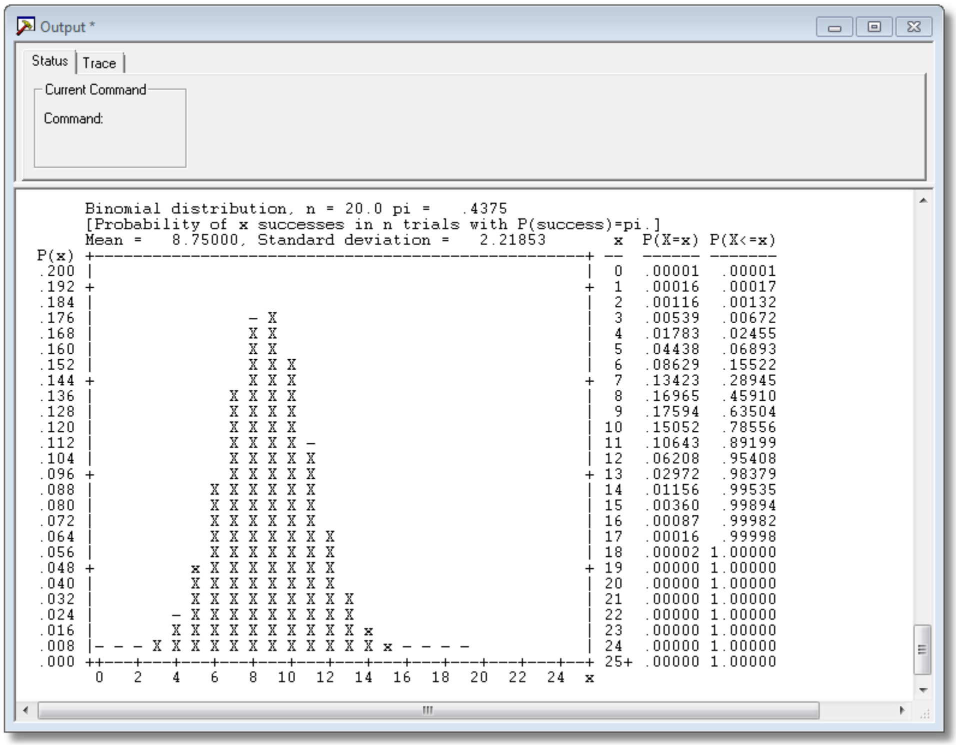
There are a variety of tools that can be used to display probability distributions. Precise, accurate figures can be drawn by plotting the values of the probability distribution. Empirical approximations to probability distributions can be obtained by drawing histograms for large random samples of the random variable.

R21.7.1 CALC Functions that Show Discrete Distributions

CALCULATE provides numerous functions for computing continuous and discrete probabilities and densities from a variety of distributions. The following additional functions will produce tables and character based plots for discrete distributions:

Tbb(*p*,*n*)	for binomial probabilities with probability *p*, *n* trials,
Tbp(*lambda*)	for Poisson with mean *lambda*,
Tbg(*p*)	for geometric with parameter *p*,
Tbn(*p*,*n*)	for negative binomial with probability *p* and *n* successes,
Tbh(*p*,*m*,*n*)	for hypergeometric with probability *p*, population size *m*, and *n* successes.

Calculating the function with specified parameters produces the listing and figure, as shown in the illustration below. **CALC ; List ; Tbb(.4375,20) $** produces the following results in the output window:



```
     Binomial distribution, n = 20.0 pi =    .4375
     [Probability of x successes in n trials with P(success)=pi.]
     Mean =   8.75000, Standard deviation =   2.21853        x  P(X=x) P(X<=x)
P(x) +---------------------------------------------------+ --  ------ -------
.200 |                                                   |  0  .00001  .00001
.192 +                                                   +  1  .00016  .00017
.184 |                                                   |  2  .00116  .00132
.176 |                - X                                |  3  .00539  .00672
.168 |                X X                                |  4  .01783  .02455
.160 |                X X                                |  5  .04438  .06893
.152 |                X X X                              |  6  .08629  .15522
.144 +                X X X                              +  7  .13423  .28945
.136 |              X X X X                              |  8  .16965  .45910
.128 |              X X X X                              |  9  .17594  .63504
.120 |              X X X X                              | 10  .15052  .78556
.112 |              X X X X -                            | 11  .10643  .89199
.104 |              X X X X X                            | 12  .06208  .95408
.096 +              X X X X X                            + 13  .02972  .98379
.088 |            X X X X X X                            | 14  .01156  .99535
.080 |            X X X X X X                            | 15  .00360  .99894
.072 |            X X X X X X                            | 16  .00087  .99982
.064 |            X X X X X X X                          | 17  .00016  .99998
.056 |            X X X X X X X                          | 18  .00002 1.00000
.048 +          x X X X X X X X                          + 19  .00000 1.00000
.040 |          X X X X X X X X                          | 20  .00000 1.00000
.032 |          X X X X X X X X X                        | 21  .00000 1.00000
.024 |        - X X X X X X X X X                        | 22  .00000 1.00000
.016 |        X X X X X X X X X X x                      | 23  .00000 1.00000
.008 |- - - X X X X X X X X X X X X x - - - -            | 24  .00000 1.00000
.000 ++---+---+---+---+---+---+---+---+---+---+---+---+--+ 25+ .00000 1.00000
      0   2   4   6   8  10  12  14  16  18  20  22  24  x
```

Figure R21.5 Binomial Distribution with CALC Functions

R21.7.2 Plotting a Density

You can plot any function using the **FPLOT** command. One use might be to plot a known probability density function. The form of the **FPLOT** command is

```
SAMPLE      ; 1 $
FPLOT       ; Fcn = the specification of the function as a function of x
            ; Labels = x
            ; Plot(x)
            ; Start = some value in the range of x
            ; Limits = the range of variation of the variable
            ; Pts = the number of points to plot (and connect) $
```

You may also add a title to the figure with **; Title = the desired title**. Note that the function need not simply be a function of *x*; it can involve other parameters as well. You might, for example, use

```
CALC        ; mu = a value ; sigma = a value $
PROC $
FPLOT       ; ... ; Fcn = (1/sigma) * N01((x - mu)/sigma) / Phi((a - mu)/sigma)
            ; ... $
ENDPROC $
EXECUTE     ; a = 0, 1.5, .5 $
```

to plot the density of a truncated normal variable as a function of the mean, *mu*, the standard deviation, *sigma*, and the upper truncation point, *a*.

The **SAMPLE ; 1 $** command is used here because **FPLOT** will compute the sum over the current sample of whatever function is specified. This will allow you to plot log likelihood functions. For the purpose here, you do not wish to sum, so you reduce the sample to just one observation. The example below plots the density of the logistic random variable.

```
SAMPLE      ; 1 $
FPLOT       ; Labels = x
            ; Plot(x)
            ; Limits = -4,4
            ; Start = 0
            ; Pts = 100
            ; Fcn = Exp(x) / (1 + Exp(x))^2
            ; Title = Logistic Distribution $
```

```
SAMPLE      ; 1 $
FPLOT       ; Labels = x
            ; Plot(x)
            ; Limits = -4,4
            ; Start = 0
            ; Pts = 100
            ; Fcn = Exp(x) / (1 + Exp(x))^2
            ; Title = Logistic Distribution $
```

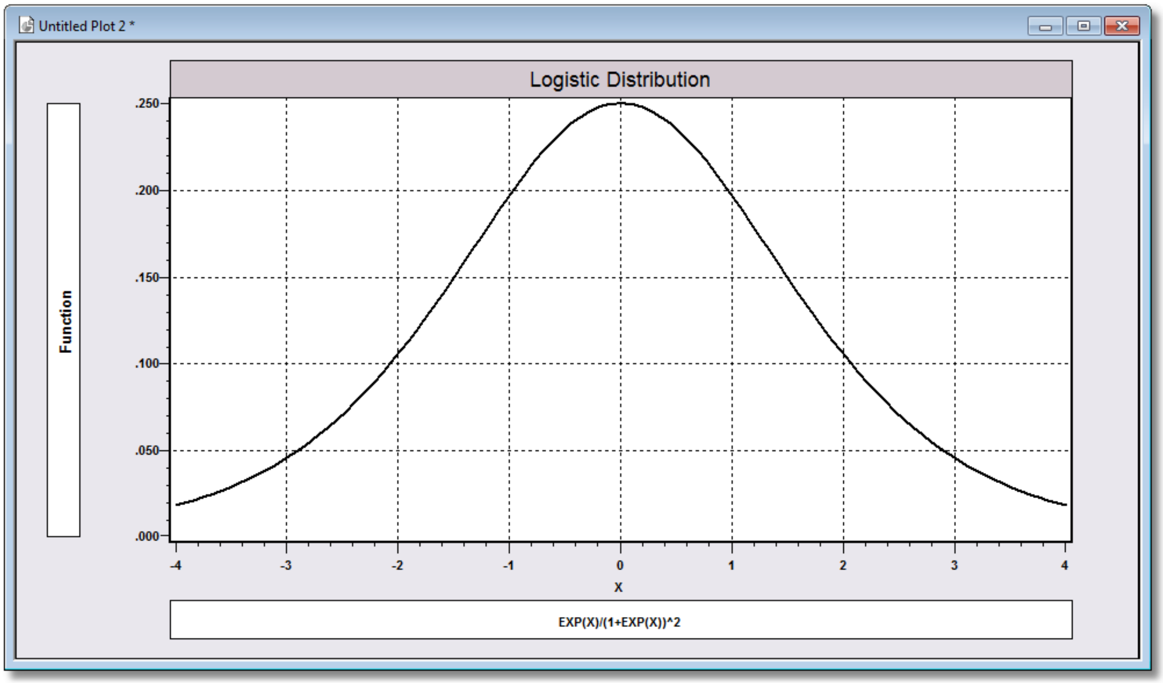


Figure R21.6 Density of the Logistic Distribution

R21.7.3 Drawing a Distribution by Plotting a Histogram

The **HISTOGRAM** command provides sufficient resolution to produce a reasonable estimate of a distribution if the sample is large enough. You need only draw your random sample using any of the methods discussed earlier, then use **HISTOGRAM** to plot the distribution. For example, the application below displays an empirical estimate of the density of a mixture of normal distributions, with even mixing of two reasonably widely separated normal distributions:

```
ROWS       ; 10000 $                          Add rows to data area
CALC       ; Ran(1234567) $                   Make exercise replicable
SAMPLE     ; 1-10000 $                        Use a large sample
CREATE     ; x1 = Rnn(1,1) ; x2 = Rnn(5,1) $  Two normal distributions
CREATE     ; u = Rnd(2) - 1 $                 Mixture, 50/50
CREATE     ; v = u*x1 + (1-u)*x2 $            The mixed variables
HISTOGRAM ; Rhs = v ; Int = 60 ?              60 bar histogram
           ; Title = Mixture of Two Normal Distributions $
```

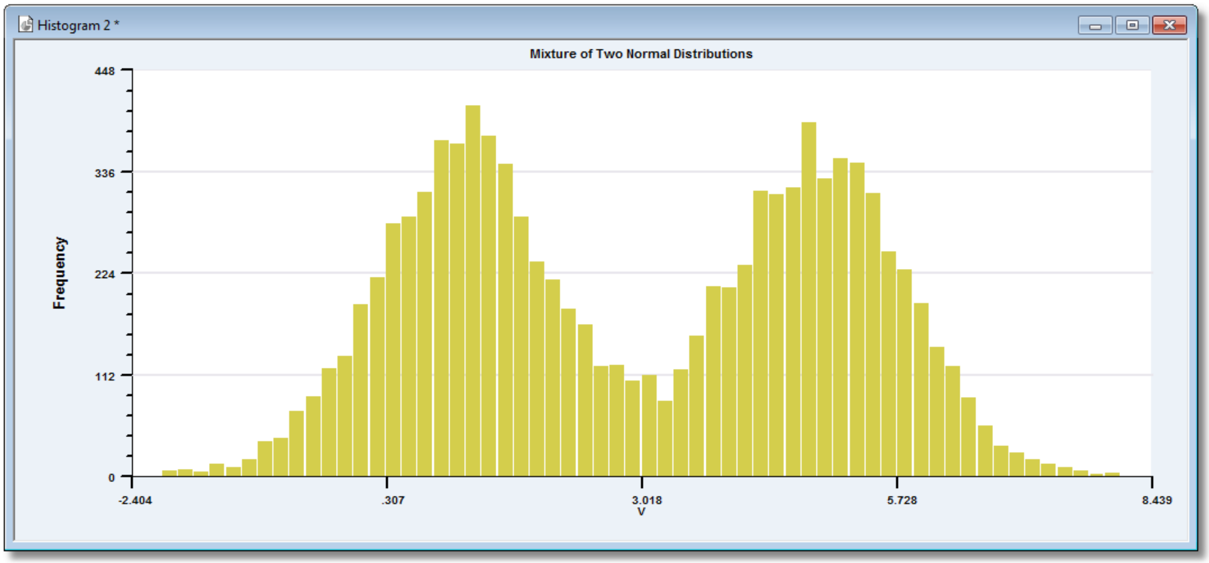


Figure R21.7 Histogram for Draws from a Bimodal Distribution

R21.7.4 Sampling Experiments

As the preceding illustrates, the random number generators combined with the **HISTOGRAM** command can be used to produce an empirical estimate of a distribution (density). This device could also be used to demonstrate the generation of data through data generating processes. Thus, events can be simulated in this fashion. The following (admittedly pretty basic) application illustrates a coin tossing experiment. Each player tosses a fair coin 10 times. What does the distribution of the number of heads tossed look like? (We know this is binomial, but we simulate it the hard way anyway.)

First, set the number of players.

```
CALC          ; players = ... $        How many players?
SAMPLE        ;  1 - players $
```

Now, toss 10 coins, coding 0 for tails, 1 for heads, and add the 1s.

```
SAMPLE        ;  1-100 $
CALC          ;  Ran(12345) $
CREATE        ;  toss1   = Rnd(2) - 1  ? (of course, there is an easier way)
              ;  toss2   = Rnd(2) - 1  ;  toss3 = Rnd(2) - 1
              ;  toss4   = Rnd(2) - 1  ;  toss5 = Rnd(2) - 1
              ;  toss6   = Rnd(2) - 1  ;  toss7 = Rnd(2) - 1
              ;  toss8   = Rnd(2) - 1  ;  toss9 = Rnd(2) - 1
              ;  toss10 = Rnd(2) - 1 $
CREATE        ;  heads = toss1 + toss2 + toss3 + toss4 + toss5 +
                         toss6 + toss7 + toss8 + toss9 + toss10 $
HISTOGRAM ; Rhs    = heads ; Title = Histogram for Number of Heads $
```

The following shows the result of this simulation with 100 players.

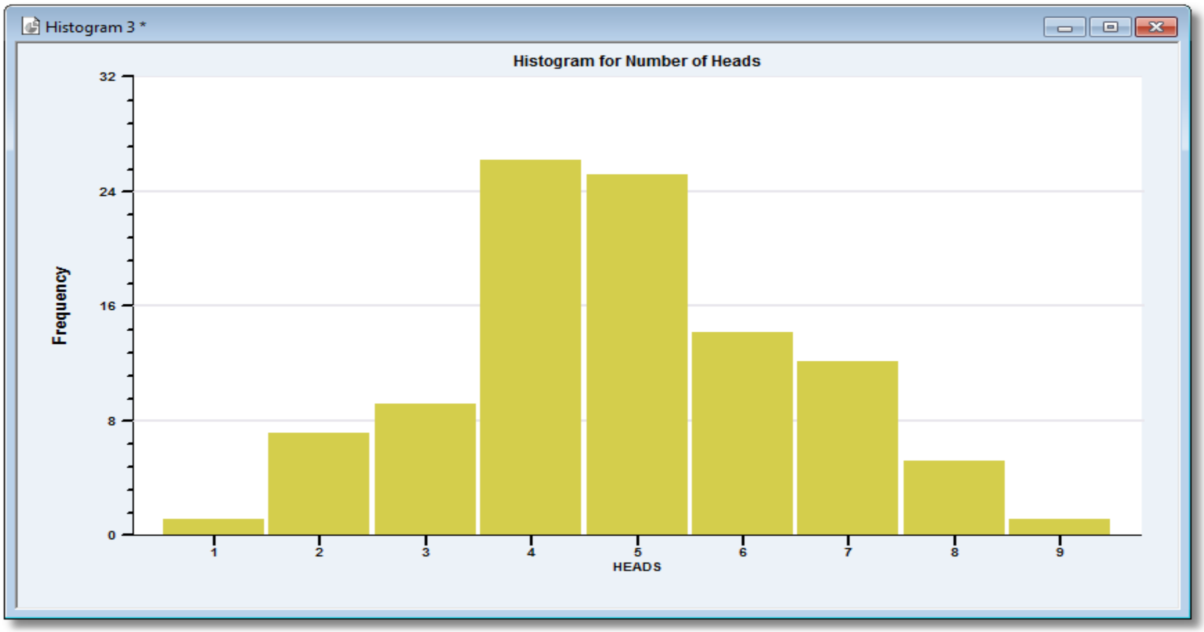


Figure R21.8 Sampling Experiment for Coin Tosses

R21.7.5 The Law of Large Numbers and the Central Limit Theorem

Demonstrating the central limit theorem always takes a bit of creativity. The following shows one approach. The procedure shows the effect of increasing *n* on the observed frequency distribution of sample means of *n* observations drawn from some decidedly nonnormal distribution. This program could be modified to compare more than three sample sizes, and can be changed easily to apply the result to sampling from any desired distribution. The annotations describe where modifications should be made. These commands create columns of sample means of three, 10 and 25 observations.

```
SAMPLE        ; 1-10000 $ desired number of means
CALC          ; Ran(1234567) $
CREATE        ; n1 = Trn(3,0) ; n2 = Trn(10,0) ; n3 = Trn(25,0) ; row = Trn(1,1) $
```

This line can be changed to use a different parent distribution

```
CREATE        ; c2 = Rnx(2) $
MATRIX        ; xb3 = Gxbr(c2,N1) ; xb10 = Gxbr(c2,n2) ; xb25 = Gxbr(c2,n3) $
```

To show the force of the result, we now produce histograms for our samples of means, using the same scale for all three figures. The limits are taken from the one known to have the greatest range of variation.

```
SAMPLE      ; 1-400 $
CREATE      ; mean1 = xb3(row)  ; mean2 = xb10(row) ; mean3 = xb25(row) $
CALC        ; a0 = Min(mean1) ; a1 = Max(mean1) $
CREATE      ; z1 = Sqr(3)*(mean1-2)/Sqr(2) ; z2 = Sqr(10)*(mean2-2)/Sqr(2)
            ; z3 = Sqr(25)*(mean3-2)/Sqr(2) $
```

These three histograms demonstrate the operation of the law of large numbers.

```
HISTOGRAM ; Rhs  = mean1 ; Int = 60 ; Limits = a0,a1
          ; Title = Means of Samples of 3 $
HISTOGRAM ; Rhs  = mean2 ; Int = 60 ; Limits = a0,a1
          ; Title = Means of Samples of 10 $
HISTOGRAM ; Rhs  = mean3 ; Int = 60 ; Limits = a0,a1
          ; Title = Means of Samples of 25 $
```

These three histograms demonstrate the force of the central limit theorem.

```
HISTOGRAM ; Rhs = z1 ; Int = 60 ; Limits = -3,3 $
HISTOGRAM ; Rhs = z2 ; Int = 60 ; Limits = -3,3 $
HISTOGRAM ; Rhs = z3 ; Int = 60 ; Limits = -3,3 $
```

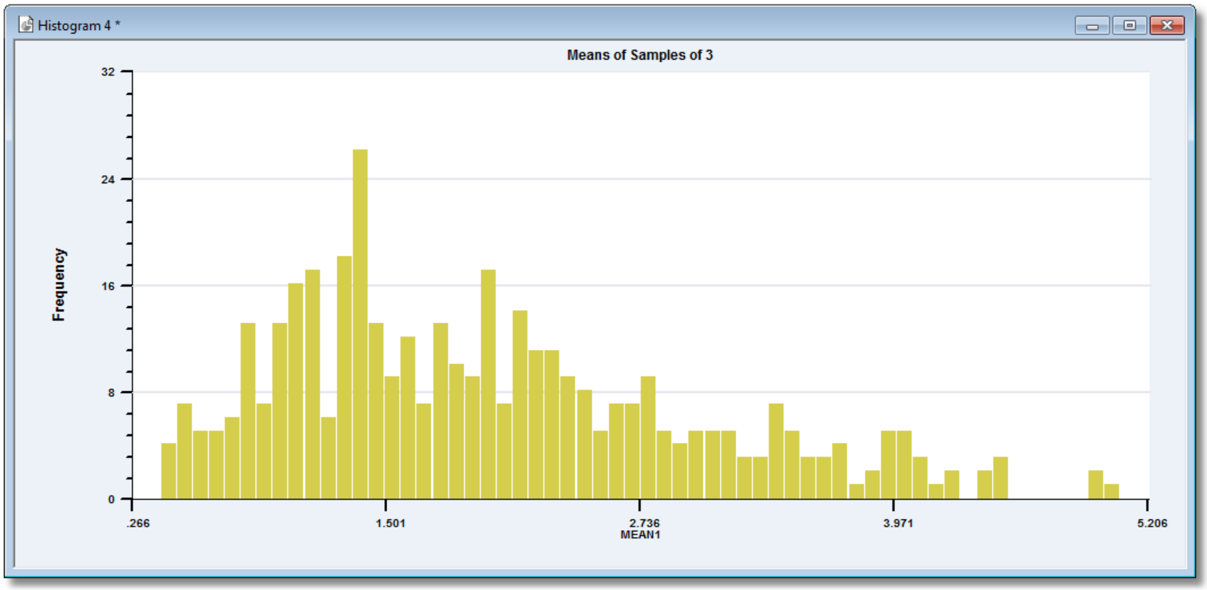


Figure R21.9a Means of Samples of 3

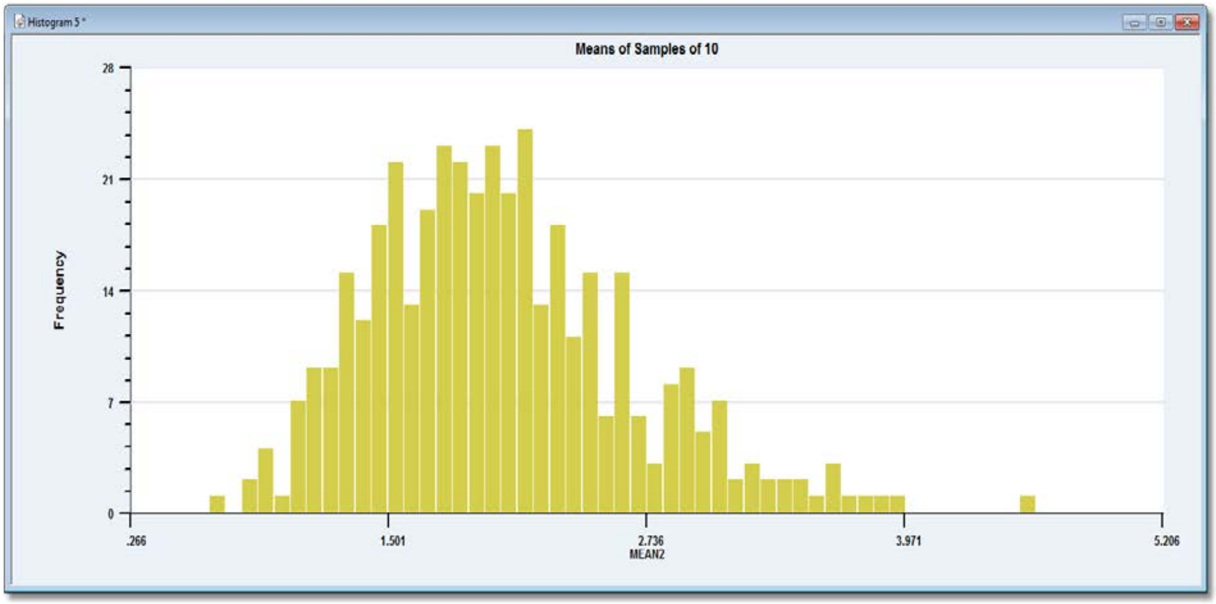


Figure R21.9b Means of Samples of 10

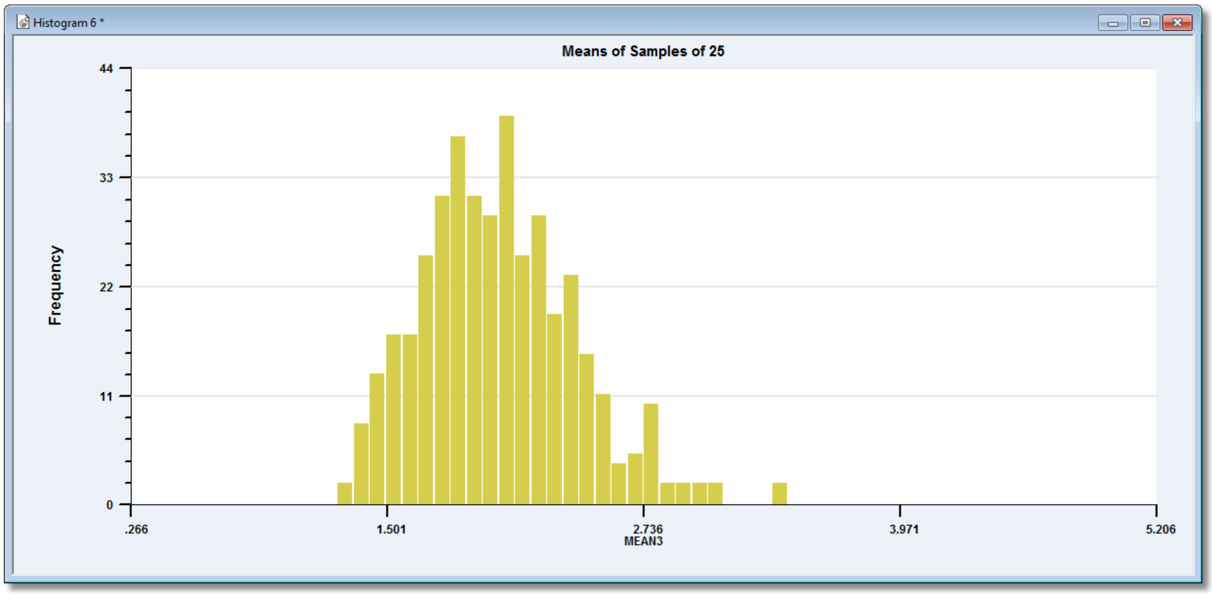


Figure R21.9c Means of Samples of 25

R21.8 Urn Experiments

The final set of applications we consider is a class of sampling experiments known as 'urn experiments.' These can become extremely elaborate. *LIMDEP* is well suited to make even the most complicated of these quite simple to program. We consider a basic template to carry out the following two steps:

Step 1. An 'urn' is initialized with a fixed mixture of n_b blue and n_r red balls. (This can be generalized to more colors without great difficulty.) Then, $n_0 = n_b + n_r$.

Step 2. For draws $i = n_0 + 1$ to *ntotal*, we reach into the urn and draw one of the (i - 1) balls then in the urn. Depending on the color drawn, add another ball to the urn. The new ball added may be blue or red, depending on some decision rule. At each draw *i*, watch the behavior of the proportion of red balls in the urn.

The following program can be used to carry out these steps. The only changes needed in the program to accommodate a particular experiment are the setting of *n*0, *nblue*, and *ntotal* at the beginning to dictate the parameters of the experiment and the augmentation rule for adding the next ball (i.e., the **CALC** command which creates newball). The program carries out the entire experiment and displays the proportion of red balls by plotting it.

The application that follows the program is the 'Polya' experiment. In this experiment, the augmentation rule is to add a ball that is the same color as the one drawn. In this experiment, there is no fixed limiting value for the proportion. It depends entirely on the initial conditions and on the first few balls drawn. An interesting aspect of this experiment can be seen by starting it from the same initial conditions with a relatively small *n*0.

First, set up the initial conditions: The *n*0 in the following command is the number of 'balls' to be in the urn at the outset. The **CREATE** command creates the initial balls, with, say, 'blue' coded 0 and 'red' coded 1. This then starts the experiment with a fixed number of blue and red balls in the urn.

```
CALC      ; n0 = ... your setting ; nblue = ... ; ntotal = ... $
SAMPLE    ; 1 - n0 $
CREATE    ; ball = 0*Ind(1,nblue) + 1*Ind((nblue+1),n0) $ nred = n0 - nblue
```

Now, do the experiment of adding balls to the urn according to some rule and, at each step, compute the appropriate descriptive statistics. *ntotal* = the number of balls that will be in the urn at the end of the experiment. Generally urn experiments involve drawing a ball from the existing stock in the urn, then adding one to the urn by a prescribed rule. We create a variable which at observation i points randomly to one of the balls that is already in the urn. For *i* = *n*0+1,... *pick* is a random integer from the values 1,2,...,*i*-1.

```
SAMPLE    ; 1 - ntotal $             Total number of balls at the end of the experiment.
CREATE    ; i = Trn(1,1) - 1 $       Creates i = 0,1,2,... for the whole sample.
CREATE    ; If (_obsno > n0) pick = Rnd(i) $
```

We will now create a procedure that will generate a Markov chain of balls added to the urn. The **CREATE** command carries out the rule for adding a new ball.

```
PROC $
CALC        ; row = pick(obs)        ? CALC picks from the existing balls for
            ; oldball = ball(row)    ? observation i. Oldball gives the ball picked.
            ; newball = ... some function of oldball $
CREATE      ; ball(obs) = newball $     Add the ball at this observation.
ENDPROC $
```

For example, if red, toss a coin and add blue if heads, red if tails. Thus, if the oldball is blue = 0, add nothing. If oldball is red = 1, add a random draw, 0 or 1. This does it.

```
CALC        ; If [oldball = 1] newball = oldball * (Rnd(2)-1) $
CALC        ; n1 = n0 + 1 $
EXEC        ; obs = n1,ntotal $      Fill all cells of the column.
```

We now compute the proportion of red balls at each observation. This is done for each just by averaging the balls from 1 to *i*. This is a ‘partial sum’ computed using the recursion, **sum(i) = sum(i-1) + x(i)**.

```
SAMPLE      ; 1 - ntotal $
CREATE      ; If (_obsno = 1) sum = ball ; (Else) sum = sum[-1] + ball $
CREATE      ; mean = sum/(i+1) $
```

We now have a column of means where at observation *i*, the mean is based on observations 1 to *i*. Presumably, this column of means will converge to something, so we investigate by describing or plotting.

```
SAMPLE      ; n1 - ntotal $
PLOT        ; Lhs = i ; Rhs = mean ; Fill ; Grid ; Endpoints = 0,ntotal $
```

This program carries out the Polya experiment. We begin by placing 10 balls in the urn, three blue ones and seven red ones. We then run the experiment for 500 draws in total. Set the parameters for this application.

```
CALC        ; n0 = 10 ; nblue = 3 ; ntotal = 500 $
```

Set the initial conditions.

```
SAMPLE      ; 1 - n0 $
CREATE      ; ball = 0*Ind(1,nblue) + 1*Ind((nblue+1),n0) $ nred = n0 - nblue
SAMPLE      ; 1 - ntotal $           Total number of balls at the end of the experiment.
CREATE      ; i = Trn(1,1) - 1 $     Creates i = 0,1,2,... for the whole sample.
```

Run the experiment.

```
CREATE      ; If (_obsno > n0) pick = Rnd(i) $
PROC $
CALC        ; row = pick(obs)       ? CALC picks from the existing balls for
            ; oldball = ball(row)   ? observation i. Oldball is the ball picked.
            ; newball = oldball $
CREATE      ; ball(obs) = newball $    Add the same ball as picked.
ENDPROC $
CALC        ; n1 = n0 + 1 $
EXEC        ; obs = n1,ntotal $        Fill all cells of the column.
```

Compute and display the results.

```
SAMPLE      ; 1 $
CREATE      ; sum = ball $
SAMPLE      ; 1 - ntotal $
CREATE      ; If (_obsno > 1) sum = sum[-1] + ball $
CREATE      ; mean = sum/(i+1) $
SAMPLE      ; n1 - ntotal $
PLOT        ; Lhs = i
            ; Rhs = mean ; Fill
            ; Grid ; Endpoints = 0,ntotal $
```

We ran this experiment a second time with the same initial conditions, but a different set of picks by saving the second set of results in a variable named *mean*2, then plotting both sets of results in the same figure. The result is shown in the second figure.

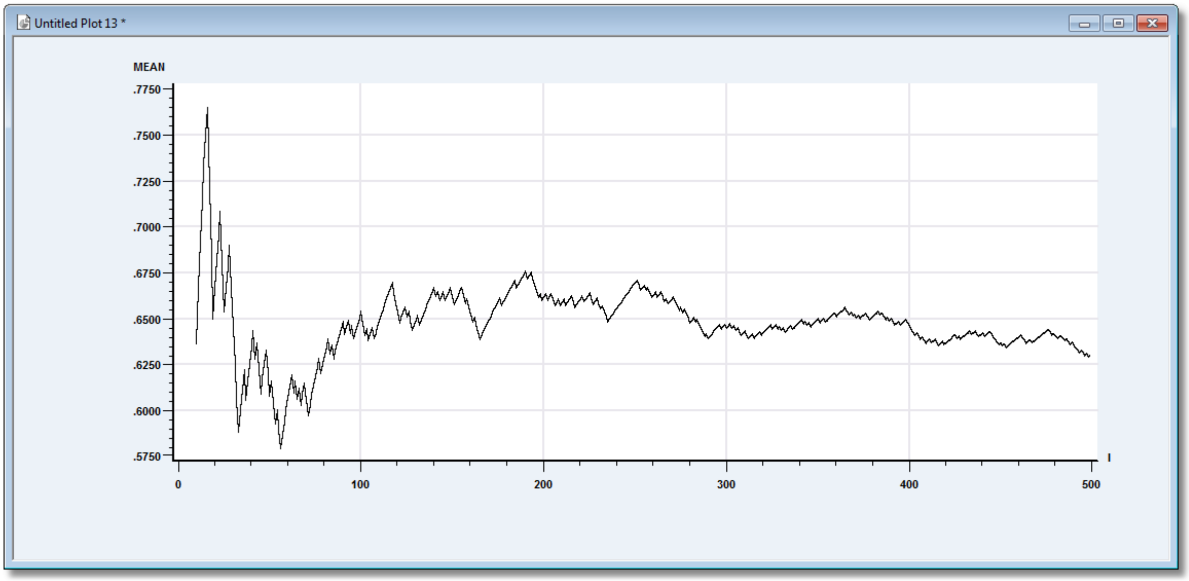


Figure R21.10a Polya Experiment

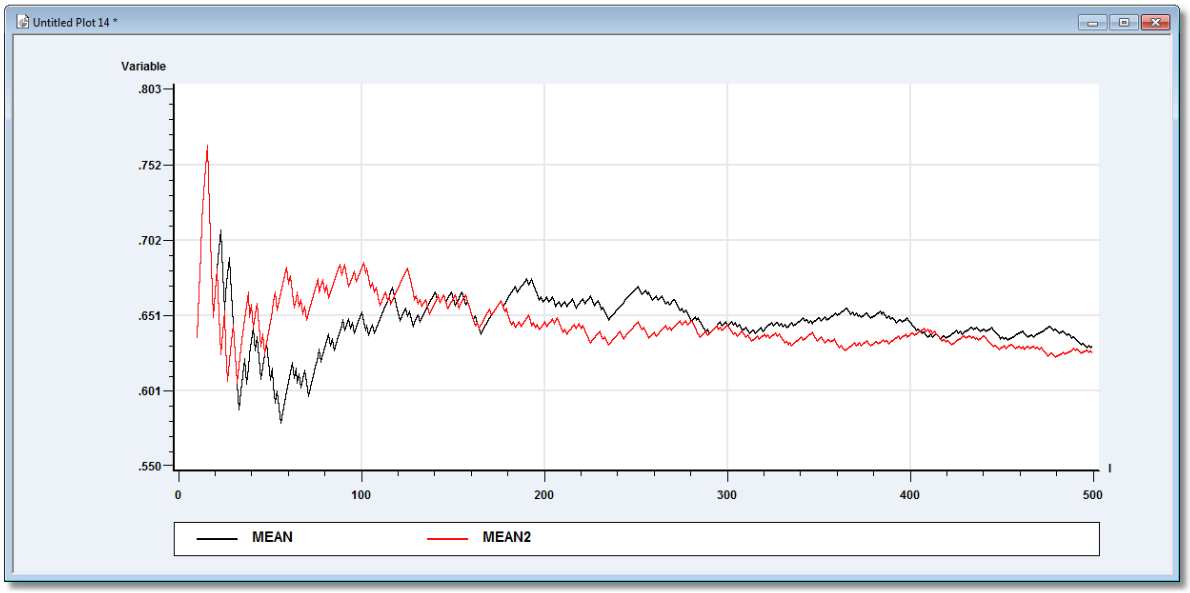


Figure R21.10b Two Runs of the Polya Experiment

R22: Models for Panel Data

R22.1 Introduction

This chapter will introduce *LIMDEP*'s collection of programs for analysis of panel data. *LIMDEP* supports by far a larger variety of model formulations for panel data than any other package. Nearly all of the models supported by the program, including dozens of linear and nonlinear specifications, provide special treatments for panel data including fixed and random effects, stratification, latent class models, random parameters, multilevel effects, and 'cluster' corrections for layered and stratified data sets. This goes far beyond the familiar fixed and random effects linear regression, Poisson and logit models typically found elsewhere, and includes sample selection models, multinomial logit, probit, censored and truncated regression, and a large variety of loglinear models.

The methods used for estimation of these models have many common features, but will also vary a bit from one application to the next. This chapter will summarize the background theory and practical aspects of the estimation methods for *LIMDEP*'s panel data models. Subsequent chapters in the *Econometric Modeling Guide* will then describe the specific models and the *LIMDEP* commands used to estimate them.

Section R22.5 will describe the basic forms of the commands for these models and the results that each will produce. There are additional features and options with each of the three broad groups, fixed effects, random effects and latent class models. Chapters R23-R25 will give technical details as well as additional model commands and features for specific model classes, common (fixed and random) effects, random parameters and latent class models.

NOTE: Users of these model forms and programs should consult Chapters R4 and R5 for discussion of data sets for panel data models in *LIMDEP*.

R22.2 Panel Data Models

Panel data treatments in *LIMDEP*, broadly defined, are those models and specifications that directly use the information that observations are grouped either because of common membership in a class (cluster and strata, for example) or because the observations constitute multiple observations on the same entity (person, firm, country). A not quite complete list of the set of programs that contain estimators and model forms for these treatments includes the following basic forms that support various arrangements of fixed and random effects:

- linear regression,
  - fixed and random effects models,
  - random coefficients models,
  - heteroscedasticity and autocorrelation,
  - time series cross section (TSCS) and SURE models,
  - nested random effects,
  - simultaneous equations models,
  - dynamic panel data models,
- nonlinear regression with exponential conditional mean,
- binary logit, probit, Gompertz, complementary log log and arctangent,
- bivariate probit models and sample selection models for binary choice,

- ordered probit, logit Gompertz and complementary log log,
- generalized ordered probit models and ordered probit models with selection,
- tobit, censored and truncated regression models,
- exponential regression model,
- Weibull, gamma, binomial, geometric, inverse Gaussian, and other loglinear models,
- Poisson and negative binomial regression models,
- Nonpoisson count models for over and underdispersion,
- stochastic frontier models,
- survival models – parametric models with time varying covariates,
- sample selection models,
- multinomial, multiperiod, random effects probit model.

The list will also be extended to include the large number of programs that fit random parameters, and latent class models. These include the ones listed above as well as numerous others.

LIMDEP does not require panels to be 'balanced.' Only TSCS and SURE require that there be the same number of rows of data for each individual. But, the set of observations must be 'contiguous.' That is, for all models listed above, the set of observations for a particular individual (group) must be a consecutive set of observations in the data set. Section R22.3.2 discusses an operation that can be used when the panels in the original data set are not contiguous.

NOTE: Much of the econometrics literature on panel data models focuses on the balanced panel case and treats the unbalanced panel as in inconvenient extension. This is what is necessary to keep the mathematics manageable. (See, e.g., Baltagi (2005).) However, this is a point at which theory and practice diverge. In *LIMDEP*, *all* panels are treated as unbalanced. The balanced panel is the special case, though only in a trivial way that will be invisible to you.

R22.3 Data Arrangement and Setup

Your data are assumed to consist of variables:

$$y_{it}, x_{1it}, x_{2it}, ..., x_{Kit}, I_{it}, \; i = 1,...,N, t = 1,...,T_i,$$

y_{it} = dependent variable,
$\mathbf{x}_{it}$ = set of independent variables,
I_{it} = stratification indicator,
K = number of regressors, not including *one*,
N = number of groups,
T_i = number of observations in group '*i*.'

The data set for all panel data models will normally consist of multiple observations, denoted $t = 1,...,T_i$, on each of $i = 1,...,N$ observation units, or 'groups.' A typical data set would include observations on several persons or countries each observed at several points in time, T_i, for each individual. In the following, we use '*t*' to symbolize 'time' purely for convenience. The panel could consist of *N* cross sections observed at different locations or *N* time series drawn at different times, or, most commonly, a cross section of *N* time series, each of length T_i. The estimation routines are structured to accommodate large values of *N*, such as in the national longitudinal data sets, with T_i being as large or small as dictated by the study but not directly relevant to the internal capacity of the estimator. (The size of T_i does become relevant in the two way models.)

NOTE: With the current revision of *LIMDEP*, there is no limit on the number of groups in a panel. Earlier versions had an upper limit of 20,000 groups. This enhancement will come at the cost of very slightly slower computation, but if you have less than several hundred thousand observations, the difference should not be perceptible.

We define a *balanced panel* to be one in which T_i is the same for all *i*, and, correspondingly, an *unbalanced panel* is one in which the group sizes may be different across *i*.

NOTE: Panels are never required to be 'balanced.' That is, the number of time observations, T_i may vary with '*i*.' The computation of the panel data estimators is neither simpler nor harder with constant T_i. No distinction is made internally. There are some theoretical complications, though.

R22.3.1 Data Arrangement

Data for the panel data estimators in *LIMDEP* are assumed to be arranged contiguously in the data set. Logically, you will have

$$Nobs = \sum_{i=1}^{N} T_i$$

observations on your independent variables, arranged in a data matrix

$$\mathbf{X} = \begin{bmatrix} \text{T}_1 \text{ observations for group 1} \\ \text{T}_2 \text{ observations for group 2} \\ \dots \\ \text{T}_N \text{ observations for group } N \end{bmatrix}$$

and likewise for the data on *y*, the dependent variable. When you first read the data into your program, you should treat them as a cross section with *nobs* observations. The partitioning of the data for panel data estimators is done at estimation time. Chapter R5 contains further details on how to set up and use panel data sets.

NOTE: Missing data are handled automatically by this estimator. You need not make any changes in the current sample to accommodate missing values – they will be bypassed automatically. Group sizes and all computations are obtained using only the complete observations. Whether or not you have used **SKIP** to manage missing values, this estimator will correctly arrange the complete and incomplete observations.

R22.3.2 Reordering Balanced Panels

Panel data may happen to be arranged by period rather than by group. For example, you might have data on 1,000 firms in each of 10 years, with the first 1,000 observations being year 1, the second 1,000 year 2, and so on. For nearly all panel data functions in *LIMDEP*, you will need to rearrange these data so that the first 10 observations are firm 1, the next 10 are firm 2, and so on. If you have no more than 100,000 observations in total, you can request that such a panel be reordered by specifying the following artificial linear regression command

```
REGRESS     ; Lhs = one ; Rhs = one ; Panel
            ; Pds = … or ; Str = …
            ; Reorder $
```

This reorders the entire data set so that it is properly arranged for the panel data estimators. Further details on this operation are given in Chapter R5.

R22.3.3 CREATE Commands for Panel Data

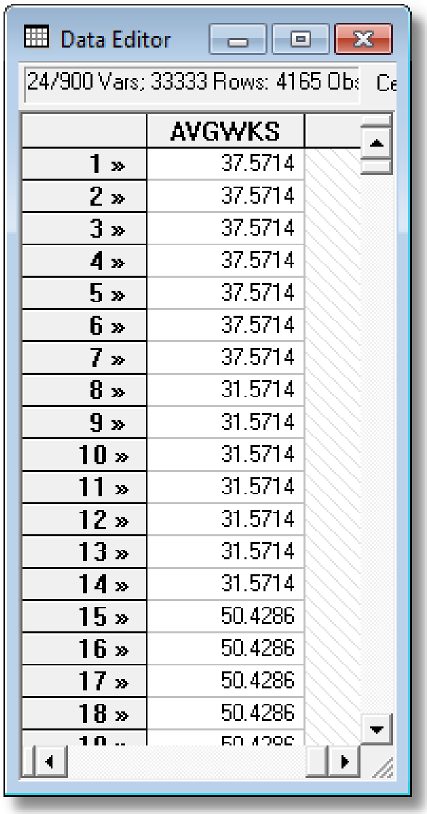
Data transformation functions and matrix operations for panel data are described in Chapters R4 and R5. One particular function that you are likely to find useful is the device to create a variable that contains the group means of a time varying variable in a panel. To create a new variable that replicates for each observation in a group the mean of that group, use the group means function,

```
      CREATE     ; z = Group Mean (variable, Str = name or number)  $
or    CREATE     ; y = Group Mean (variable, Pds = name or number) $
```

The function requires a panel data specification, precisely the same sort as used to specify panels in the model commands. The function produces a report when computed, such as

```
+------------------------------------------------------------------+
| Variable = ____________ Variable Groups    Max    Min    Average |
| AVGWKS     Group means  WKS           595      7      7        7.0 |
+------------------------------------------------------------------+
```

The variable is added to the data set, as shown in Figure R22.1.



	AVGWKS
1 »	37.5714
2 »	37.5714
3 »	37.5714
4 »	37.5714
5 »	37.5714
6 »	37.5714
7 »	37.5714
8 »	31.5714
9 »	31.5714
10 »	31.5714
11 »	31.5714
12 »	31.5714
13 »	31.5714
14 »	31.5714
15 »	50.4286
16 »	50.4286
17 »	50.4286
18 »	50.4286

Figure R22.1 Group Means of Weeks

This function must be used in isolation, not as part of another command nor in a compound function. Use a new **CREATE** command for each variable. Other available panel data functions are

Group deviations	GroupDevs	(deviations from own group means, $z_{it} - \bar{z}_i$)
Group lagged value	GroupLags	(the first observation becomes missing, $z_{i,t-1}$)
Group first difference	GroupDiff	(the first observation becomes missing, $z_{it} - z_{i,t-1}$)

These are the main functions that you will use in **CREATE** to transform panel data. A variety of other functions are listed in Section R4.3.17.

R22.4 General Model Forms for Panel Data

The class of models presented in this chapter are generically denoted

$$P(y_{it}) = g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it})$$

where

$P(.)$ = the probability density function of the observed random variable, y_{it}.

i = $1,...,N$ denotes the *i*th group or individual.

The number of groups is usually unlimited, but in a few cases is limited. This generally applies to the fixed effects models where the upper limit in some cases is 250,000. There is no limit on the number of groups in the random effects, random parameters or latent class formulations.

t = $1,...,T_i$ denotes the *t*th period, ranging from one to a person or group specific T_i. With only one exception that is dictated by the structure of the model (TSCS and SURE), *LIMDEP always* allows T_i to vary across groups. That is, panels may always be unbalanced.

y_{it} = the observed dependent variable.

$\mathbf{x}_{it}$ = is used to denote an observed vector of independent variables. This may include variables which vary across both groups and periods, and, in some applications, may also involve variables which vary across groups but are constant across periods, such as group specific dummy variables or time invariant effects such as gender in microeconometric applications.

$\boldsymbol{\beta}_i$ = the parameter vector for the *i*th individual. This may vary completely across individuals, as in the random coefficients models, or it may have a fixed component and a subvector which varies across groups, as in the usual fixed effects model. It may also be constant across groups and periods, as in the random effects model.

ε_{it} = the stochastic component of the model. The symbol is used generically to indicate the stochastic nature of the model, not necessarily a ‘disturbance.’

g = the density of the observed random variable conditioned on the arguments.

LIMDEP supports the following general model forms for panel data:

Fixed Effects Models

$$g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it}) = g(\boldsymbol{\beta}'\mathbf{x}_{it} + \alpha_i, \varepsilon_{it})$$

The 'effect' α_i is assumed to be correlated with the included variables, x_{it}. This model is generally estimated by including *N* group dummy variables in the model. Familiar results are largely based on least squares with the 'within' transformation (deviations from group means). This does not work for nonlinear models. *LIMDEP* uses maximum likelihood methods instead.

Random Effects Models

$$g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it}) = g(\boldsymbol{\beta}'\mathbf{x}_{it} , \varepsilon_{it} + u_i)$$

The effect u_i in this model is assumed to be uncorrelated with the included variables $\mathbf{x}_{it}$. Familiar results are based on generalized least squares in linear models. In nonlinear models, it is necessary to analyze the likelihood function instead.

Random Coefficients Models

$$g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it}) = g[\boldsymbol{\beta}_i(\mathbf{z}_i,\mathbf{v}_i)'\mathbf{x}_{it} , \varepsilon_{it}]$$

The individual specific coefficient vector is modeled as the result of a random process that depends on observable heterogeneity, $\mathbf{z}_i$ and unobserved heterogeneity, $\mathbf{v}_i$. There are numerous variants on this 'hierarchical' (or 'multilevel') model. The random effects model can be viewed as a random coefficients model in which only the constant term is random. But, in this modeling framework as discussed below, we view the random coefficients models much more broadly than this. First, coefficients on the other exogenous variables are allowed to be random as well. Second, the random coefficients models fit in *LIMDEP* also allow for underlying heterogeneity in the distribution of the parameters, to be modeled with other individual specific characteristics. Thus, in the formulation above, the mean of the distribution of $\boldsymbol{\beta}_i$ is allowed to vary deterministically across individuals.

Latent Class Models

$$g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it}) = \mathrm{E}_{classes}\, [g((\boldsymbol{\beta}_{class}'\mathbf{x}_{it} , \varepsilon_{it}) \mid class]$$

The latent class model characterizes the population that generates the sample more broadly than just characterizing the model. The implication for the model builder is that there is latent heterogeneity in model components, similar to the random parameters model. Here, the distribution across individuals is discrete. In the random parameters model, the variation is continuous.

There is some variation across model types regarding which of the three model forms is supported. The list in the Table R22.1 suggests the extent.

Model Class	Fixed Effects	Random Effects	Random Parameters	Latent Class
Linear Regression [a]	●	●	●	●
Nonlinear Reg. (Expon.)	●	●	●	●
Binary Choice				
Probit [a]	●	●	●	●
Logit [a]	●	●	●	●
Complementary Log Log [a]	●	●	●	●
Gompertz [a]	●	●	●	●
Bivariate Probit [b]		●	●	
Bivar. Probit Selection [b]	●	●	●	
Partial Observability [b]		●	●	
Multinomial Choice				
Multinomial Logit [c]		●	●	●
Multinomial Probit [b]		●		
Ordered Probability/All [a]	●	●	●	●
Generalized Ord. Probit		●	●	
Count Data				
Poisson Regression [a]	●	●	●	●
Negative Binomial [a]	●	●	●	●
Poisson/NegBin ZIP [b]	●	●	●	●
Loglinear Models				
Normal (Exp. Regr.) [b]	●	●	●	●
Exponential [b]	●	●	●	●
Gamma [b]	●	●	●	●
Weibull [b]	●	●	●	●
Inverse Gaussian [b]	●	●	●	●
Geometric [b]	●	●	●	●
Power [b]	●	●	●	●
Binomial [b]	●	●	●	●
Limited Dependent Variable				
Tobit [a]	●	●	●	●
Censored (Grouped) Data[a]	●	●	●	●
Truncated Regression [b]	●	●	●	●
Sample Selection [b]	●	●	●	
Survival and Frontier Models				
Weibull [b]	●	●	●	●
Exponential [b]	●	●	●	●
Loglogistic [b]	●	●	●	●
Lognormal [b]	●	●	●	●
Stochastic Frontier [a]	●	●	●	●

Table R22.1 Model Formulations with Panel Data Estimators

[a] The random effects model can be estimated by standard REM techniques (GLS, quadrature) or by the simulation method with a random parameters formulation;

[b] The random effects model can only be estimated by the simulated random parameters approach.

[c] Multinomial logit with random effects can be fit as a random parameters logit model by *NLOGIT* The latent class multinomial logit model can also be fit with *NLOGIT*.

R22.5 Model Commands

There is a small amount of variation across models in the form of the commands, but in most cases, the model formulation will be as follows:

R22.5.1 Specifying the Panel

Panels may always be balanced or unbalanced. A balanced panel has the same number of observations per group; T_i is a fixed T. An unbalanced panel allows T_i to vary across individuals. Your model command will generally contain a specification

; Pds = panel specification

which tells which of these is the case. If the panel is balanced, you will give the specific value as the specification, for example as in

; Pds = 10

If the panel is unbalanced, you will provide the name of a variable which repeats within a group the number of observations in the group, as in

; Pds = ti

For example, suppose your panel data set contained 20 observations in total, in six groups with T_i = 4, 3, 2, 4, 5, and 2 observations. Then the variable named *ti* containing the 20 values

4,4,4,4, 3,3,3, 2,2, 4,4,4,4, 5,5,5,5,5, 2,2

could be used to specify this panel.

The general command

```
SETPANEL    ; Group = id variable ; Pds = name to be used for count variable $
```

can be used to set up the group size variable and to install a procedure that will automatically keep track of the panel settings for the current sample. After you use **SETPANEL**, it is only necessary to include

; Panel

in your model command. Further details on this setting appear in Section R5.3.3.

Most of the techniques described here apply to applications involving panel data sets. However, in the case of the random parameters models, the techniques can be extended to a surprising array of cross section applications. The model does not require a panel, though in practice, the model is not strongly identified by a cross section, and results may be less than ideal. Nonetheless, **; Pds = 1** is allowable, though it is superfluous. If you do not specify **; Pds** in your command, one period is assumed.

NOTE: The fixed and random effects linear regression models also allow a stratification variable to be specified with **; Str = variable**. This provides a group identifier. This is specific to this particular model. It has the same effect as the periods specification above. But, the **; Str = variable** formulation may only be used in that setting with the **REGRESS** command. See Chapter R5 for discussion of panel data indicator variables and Chapter E16 for the panel data estimator used with **REGRESS**. For nonlinear models, **; Str = variable** is used *only* for the clustering estimator for robust covariance matrices (Sections R10.1 and E18.2.6) for specifying nested, multilevel random effects (Section E18.8).

R22.5.2 Missing Data

Missing data can sometimes confuse the panel data setups. In some cases, you must remove the observations containing missing values from the sample before issuing the estimation command. In the large panel data groupings fixed effects, random parameters and latent classes, the estimator will, itself, bypass the missing data, and your observation count can give the group sizes including the missing values. The discussion for each model framework will detail specifically how missing values are to be handled. Please note this in each case for the model you are analyzing.

R22.5.3 Model Type Specifications, Output and Saved Matrices

There are some common aspects of the major classes of panel data models. We discuss these here. Later in this chapter, we will provide more extensive details on each of these estimation classes, including some of the more specialized formats. In addition to the model specific results that are retained by each estimator, the panel data estimators each create type specific matrices that are likely to be useful. For example, if requested, the general fixed effects estimator creates a matrix named *alphafe* which contains the estimated dummy variable coefficients (up to 50,000 of them).

Fixed Effects Models

A fixed effects model will generally be specified with

```
Model       ; Lhs = ...
            ; Rhs = ...
            ; Pds = specification
            ; FEM  $
```

NOTE: With two exceptions, **; Fixed Effects** may be used instead of **; FEM**. Two models, the binary logit and the negative binomial model each support two different approaches to fixed effects estimation. For these cases, **; Fixed** invokes the Chamberlain estimator for the logit model and the Hausman, Hall and Griliches estimator for the negative binomial model. *LIMDEP*'s more general, ‘true’ fixed effects estimator is invoked with **; FEM**. The differences between the approaches are discussed in the respective chapters in the *Econometric Modeling Guide*.

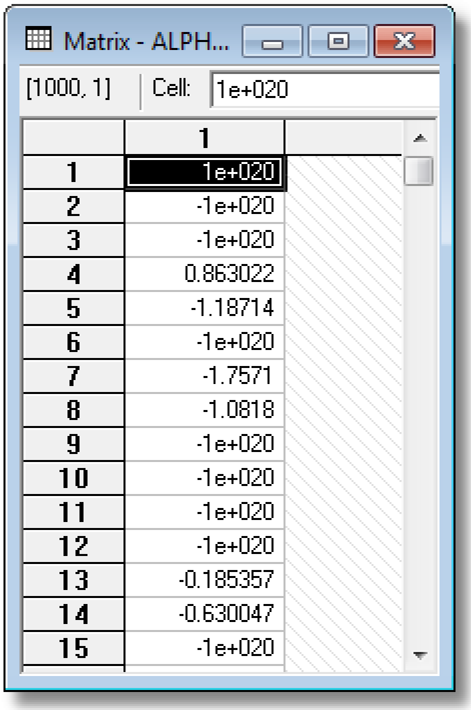
Other estimation features will usually be available as well, such as optimization parameters, fitted values, and so on. For example, the following presents estimates for a probit model with 1,000 individual intercepts using simulated data – actually 676 as 324 groups had no within group variation. See Figure R22.2 and Chapter R23 for discussion of this important point.

```
CALC          ; Ran (123457) $
SAMPLE        ; 1-5000 $
MATRIX        ; ui = Rndm(1000) $
CREATE        ; x = Rnn(0,1) ; i = Trn(5,0) $
CREATE        ; yits = -.5 + .5*x + ui(i) + Rnn(0,1) $
CREATE        ; yit = yits > 0 $
PROBIT        ; Lhs = yit ; Rhs = one,x
              ; Pds = 5
              ; FEM
              ; Partial Effects
              ; Parameters $
```

```
-----------------------------------------------------------------------------
Probit   Regression Start Values for YIT
Dependent variable                  YIT
Log likelihood function     -3154.28695
Estimation based on N =   5000, K =   2
Inf.Cr.AIC  = 6312.574 AIC/N =    1.263
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
       X|      .34595***      .01912    18.09  .0000      .30847    .38343
Constant|     -.32119***      .01850   -17.37  .0000     -.35744   -.28494
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

Nonlinear Estimation of Model Parameters
Method=Newton; Maximum iterations=100
Convergence criteria: max|dB|   .1000D-05, dF/F=  .1000D-08, g<H>g=  .1000D-08
Normal exit from iterations. Exit status=0.

-----------------------------------------------------------------------------
FIXED EFFECTS Probit Model
Dependent variable                  YIT
Log likelihood function     -1729.82433
Estimation based on N =   5000, K = 677
Inf.Cr.AIC  = 4813.649 AIC/N =     .963
Sample is  5 pds and   1000 individuals
Skipped  324 groups with inestimable ai ←
PROBIT (normal)  probability model
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
       X|      .65919***      .03125    21.10  .0000      .59795    .72043
--------+--------------------------------------------------------------------
```

```
--------+-------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
----------------------------------------------------------------------
Partial derivatives of E[y] = F[*]   with
respect to the vector of characteristics.
They are computed at the means of the Xs.
Estimated E[y|means,mean alphai]=    .435
Estimated scale factor for dE/dx=    .394
--------+-------------------------------------------------------------
        |     Partial                               Prob.      95% Confidence
     YIT|      Effect    Elasticity      z     |z|>Z*          Interval
--------+-------------------------------------------------------------
       X|     .25949***      .01523    21.04  .0000       .23531    .28366
--------+-------------------------------------------------------------
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
----------------------------------------------------------------------
```

The matrix *alphafe* contains the estimates of the dummy variable coefficients. (The large values, -1.d20 are filler for cells that correspond to inestimable coefficients. This occurs when the dependent variable in the binary choice model is always zero or one.


Matrix - ALPH...
[1000, 1] Cell: 1e+020

	1
1	1e+020
2	-1e+020
3	-1e+020
4	0.863022
5	-1.18714
6	-1e+020
7	-1.7571
8	-1.0818
9	-1e+020
10	-1e+020
11	-1e+020
12	-1e+020
13	-0.185357
14	-0.630047
15	-1e+020

Figure R22.2 Estimated Fixed Effects

Random Effects Models

Random effects models will usually be specified similarly, but as noted, there is large variation across model types. The simple specification

; Pds = specification

will suffice in a few cases, such as the stochastic frontier model, but for the probit, logit, Poisson, and others, in which there are several different panel data models, you must specify the type of model, with

; Pds = specification ; Random Effects

For example, the following small experiment estimates a (properly specified – the data exactly obey the assumptions) random effects probit model. Note that a full set of results is produced for the model, as would be in a cross section.

```
CALC      ; Ran (123457) $
SAMPLE    ; 1-5000 $
MATRIX    ; ui = Rndm(1000) $
CREATE    ; x = Rnn(0,1) ; i = Trn(5,0) $
CREATE    ; yits = -.5 + .5*x + ui(i) + Rnn(0,1) $
CREATE    ; yit = yits > 0 $
PROBIT    ; Lhs = yit ; Rhs = one,x
          ; Pds = 5
          ; Random Effects
          ; Partial Effects
          ; Parameters $
```

```
-----------------------------------------------------------------------------
Random Effects Binary Probit Model
Dependent variable                 YIT
Log likelihood function     -2838.62648
Restricted log likelihood   -3154.28695
Chi squared [   1 d.f.]       631.32094
Significance level               .00000
Estimation based on N =   5000, K =   3
Inf.Cr.AIC  = 5683.253 AIC/N =    1.137
Sample is  5 pds and  1000 individuals.
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    -.44895***      .03927   -11.43  .0000     -.52591   -.37199
       X|     .50151***      .02536    19.78  .0000      .45181    .55122
     Rho|     .49459***      .02209    22.39  .0000      .45131    .53788
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with
respect to the vector of characteristics
They are computed at the means of the Xs
Observations used for means are All Obs.

--------+--------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
     YIT|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
       X|     .13539***      .00514    18.75  .0000      .12124    .14954
--------+--------------------------------------------------------------------
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Random Parameters

The random parameters models are all specified the same way. The model command is

```
Model    ; Lhs = ...
         ; Rhs = ...
         ; Pds = specification (this is optional here)
         ; RPM (for random parameters model)
         ; Fcn = variable name (distribution) ... $
```

(There are numerous optional specifications for this model class. These are detailed in Chapter R24.) The **; Fcn = name (distribution)** specifies the variable in the Rhs list whose coefficient is random – they need not all be. The basic distributions available are normal, uniform, and triangular, which you would specify as

; Fcn = name(n) or (u) or (t)

(There are a variety of ways to modify and extend this model.) There are several other parameters that are part of this specification. Estimation is done by simulation, so you may specify the number of replications for the simulation with

; Pts = the desired number of replications

The default is 100, but you may give any value you like.

The parameters specified to be random are assumed in the preceding to be uncorrelated, so that for each one, the estimates consist of the mean of the distribution and a scale factor for the random term. You can fit a model in which the random parameters are allowed to be freely correlated by adding

; Correlation

to the model command.

The following simulates and fits a random parameters probit model in an exercise similar to the earlier examples. (Some output is omitted.)

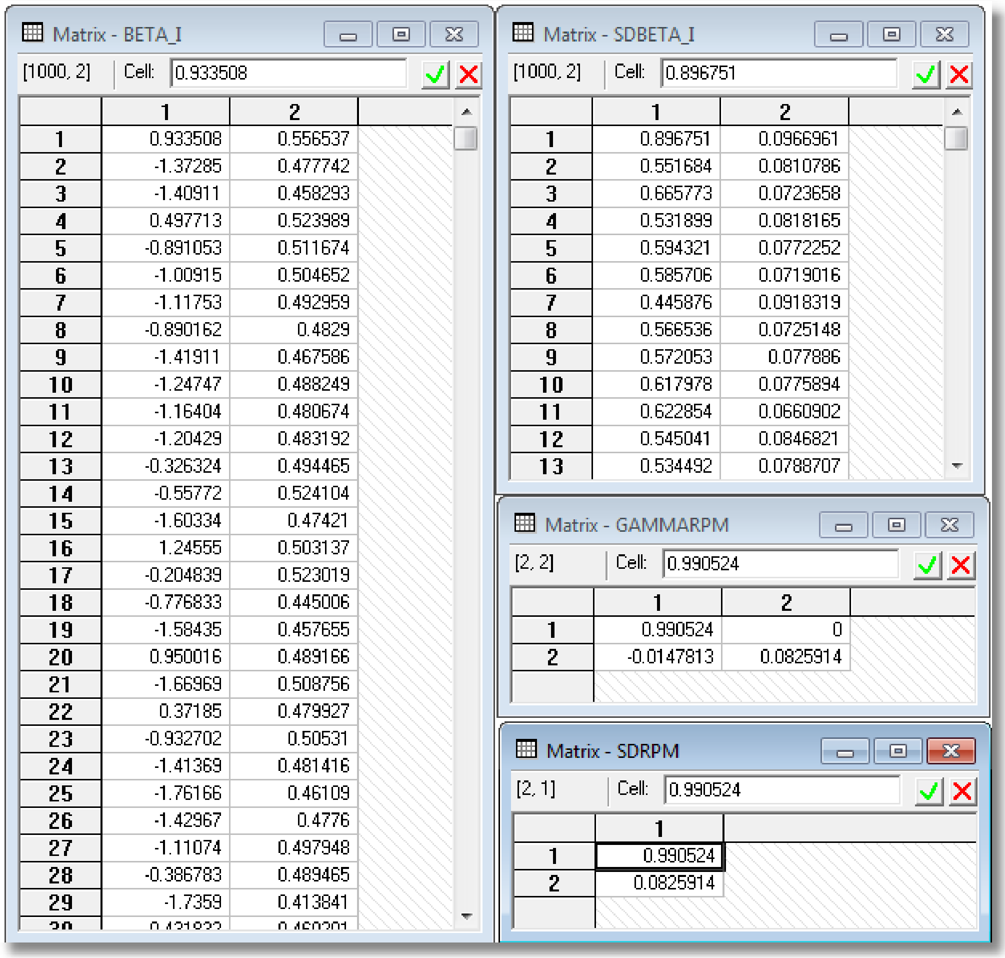
```
CALC        ; Ran (123457) $
SAMPLE      ; 1-5000 $
MATRIX      ; ui = Rndm(1000) $
CREATE      ; x = Rnn(0,1) ; i = Trn(5,0) $
CREATE      ; yits = -.5 + .5*x + ui(i) + Rnn(0,1) $
CREATE      ; yit = yits > 0 $
PROBIT      ; Lhs = yit ; Rhs = one,x
            ; Pds = 5
            ; RPM ; Fcn = one(n),x(n)
            ; Correlated ; Pts = 50
            ; Parameters $
```

```
-----------------------------------------------------------------------------
Random Coefficients  Probit   Model
Dependent variable                  YIT
Log likelihood function     -2853.67087
Restricted log likelihood   -3154.28695
Chi squared [   3 d.f.]       601.23217
Significance level               .00000
Estimation based on N =   5000, K =   5
Sample is  5 pds and   1000 individuals
PROBIT (normal)  probability model
Simulation based on  50 random draws
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Means for random parameters
Constant|    -.45179***      .02276   -19.85  .0000     -.49640   -.40718
       X|     .49366***      .02391    20.65  .0000      .44680    .54053
        |Diagonal elements of Cholesky matrix
Constant|     .99052***      .02798    35.40  .0000      .93568   1.04537
       X|     .08259***      .02207     3.74  .0002      .03933    .12585
        |Below diagonal elements of Cholesky matrix
  lX_ONE|    -.01478         .02759     -.54  .5922     -.06886    .03930
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

Implied covariance matrix of random parameters

Var_Beta|             1             2
--------+----------------------------
       1|       .981138     -.0146412
       2|     -.0146412     .00703983

Implied standard deviations of random parameters

S.D_Beta|             1
--------+--------------
       1|       .990524
       2|      .0839037
```

```
Implied correlation matrix of random parameters

Cor_Beta|            1            2
--------+--------------------------
       1|      1.00000      -.176169
       2|      -.176169      1.00000
```

The **; Parameters** specification in a random parameters model creates several matrices:

beta_i	= group specific estimates of $E[\boldsymbol{\beta}_i \mid \text{data on individual } i]$
sdbeta_-i	= group specific estimates of Std.Dev$[\boldsymbol{\beta}_i \mid \text{data } i]$
gammaprm	= estimate of $\boldsymbol{\Gamma}$ matrix
sdrpm	= estimates of $\boldsymbol{\sigma}_\beta$ for random parameters.


Matrix - BETA_I

[1000, 2] Cell: 0.933508

	1	2
1	0.933508	0.556537
2	-1.37285	0.477742
3	-1.40911	0.458293
4	0.497713	0.523989
5	-0.891053	0.511674
6	-1.00915	0.504652
7	-1.11753	0.492959
8	-0.890162	0.4829
9	-1.41911	0.467586
10	-1.24747	0.488249
11	-1.16404	0.480674
12	-1.20429	0.483192
13	-0.326324	0.494465
14	-0.55772	0.524104
15	-1.60334	0.47421
16	1.24555	0.503137
17	-0.204839	0.523019
18	-0.776833	0.445006
19	-1.58435	0.457655
20	0.950016	0.489166
21	-1.66969	0.508756
22	0.37185	0.479927
23	-0.932702	0.50531
24	-1.41369	0.481416
25	-1.76166	0.46109
26	-1.42967	0.4776
27	-1.11074	0.497948
28	-0.386783	0.489465
29	-1.7359	0.413841

Matrix - SDBETA_I

[1000, 2] Cell: 0.896751

	1	2
1	0.896751	0.0966961
2	0.551684	0.0810786
3	0.665773	0.0723658
4	0.531899	0.0818165
5	0.594321	0.0772252
6	0.585706	0.0719016
7	0.445876	0.0918319
8	0.566536	0.0725148
9	0.572053	0.077886
10	0.617978	0.0775894
11	0.622854	0.0660902
12	0.545041	0.0846821
13	0.534492	0.0788707

Matrix - GAMMARPM

[2, 2] Cell: 0.990524

	1	2
1	0.990524	0
2	-0.0147813	0.0825914

Matrix - SDRPM

[2, 1] Cell: 0.990524

	1
1	0.990524
2	0.0825914

Figure R22.3 Matrix Results from Random Parameters Model

NOTE: The random coefficients model with only the constant term specified to be random is equivalent to a random effects model as discussed earlier. However, these will not give the same answer for a given data set because the random effects model will generally be fit by full ML, using Hermite quadrature to integrate the random effect out of the terms in the log likelihood function while the random parameters (random constant) is estimated by simulation methods. If the model is correctly specified, these two estimators should produce similar answers. The larger is the sample, the closer they will be. The example below demonstrates using the artificial data generated earlier.

```
CALC        ; Ran (123457) $
SAMPLE      ; 1-5000 $
MATRIX      ; ui = Rndm(1000) $
CREATE      ; x = Rnn(0,1) ; i = Trn(5,0) $
CREATE      ; yits = -.5 + .5*x + ui(i) + Rnn(0,1) $
CREATE      ; yit = yits > 0 $
PROBIT      ; Lhs = yit ; Rhs = one,x
            ; Pds = 5
            ; RPM ; Fcn = one(n) ; Pts = 100 $
PROBIT      ; Lhs = yit ; Rhs = one,x
            ; Random Effects $
```

```
-----------------------------------------------------------------------------
Random Coefficients  Probit   Model
Dependent variable                 YIT
Log likelihood function     -2844.30849
Restricted log likelihood   -3154.28695
Chi squared [   1 d.f.]       619.95692
Significance level               .00000
McFadden Pseudo R-squared      .0982721
Estimation based on N =   5000, K =   3
Inf.Cr.AIC  = 5694.617 AIC/N =    1.139
Sample is  5 pds and   1000 individuals
PROBIT (normal)  probability model
Simulation based on 100 random draws
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Nonrandom parameters
       X|     .49965***      .02285    21.86  .0000      .45485    .54444
        |Means for random parameters
Constant|    -.45470***      .02203   -20.64  .0000     -.49787   -.41153
        |Scale parameters for dists. of random parameters
Constant|     .99086***      .02799    35.40  .0000      .93600   1.04572
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
Random Effects Binary Probit Model
Dependent variable                    YIT
Log likelihood function     -2838.62648
Restricted log likelihood   -3154.28695
Chi squared [   1 d.f.]       631.32094
Significance level               .00000
McFadden Pseudo R-squared      .1000735
Estimation based on N =   5000, K =   3
Inf.Cr.AIC  = 5683.253 AIC/N =    1.137
Sample is  5 pds and  1000 individuals.
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    -.44895***      .03927   -11.43  .0000     -.52591   -.37199
       X|     .50151***      .02536    19.78  .0000      .45181    .55122
     Rho|     .49459***      .02209    22.39  .0000      .45131    .53788
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The slope parameters differ predictably, but are close as might be expected. The correlation parameter estimated by the REM equals σ_u^2 / (1 + σ_u^2). Computing this ratio with the value 0.99086268 given for the RPM gives a counterpart to ρ of 0.495409. The two techniques are clearly finding essentially the same point. Note, as well, that the log likelihoods are similar, but as expected, not exactly the same.

Latent Class Models

Finally the command for the latent class models has a common simple form:

```
Model    ; Lhs  = ...
         ; Rhs  = ...
         ; Pds  = specification
         ; LCM (for latent class model)
         ; Pts  = number of classes, up to 9 $
```

The small example below uses exactly the same data used in the preceding examples. In this case, although the results seem reasonable, in fact, the model is misspecified – the data do not conform to a latent class model. The estimator has done its best to partition the continuously variable parameter vector (with one nonvarying element) into a discrete distribution with three mass points. Note that the true expected value of the continuously variable constant term in the model is -0.5000. If we average the two class specific coefficient vectors with weights equal to the respective class probabilities (-.75661,.60010), the results resemble the true mean vector of (-.5,.5). The **; List** parameter requests a listing of the posterior class probabilities (defined below) and the actual commands. The example shows the first few observations.

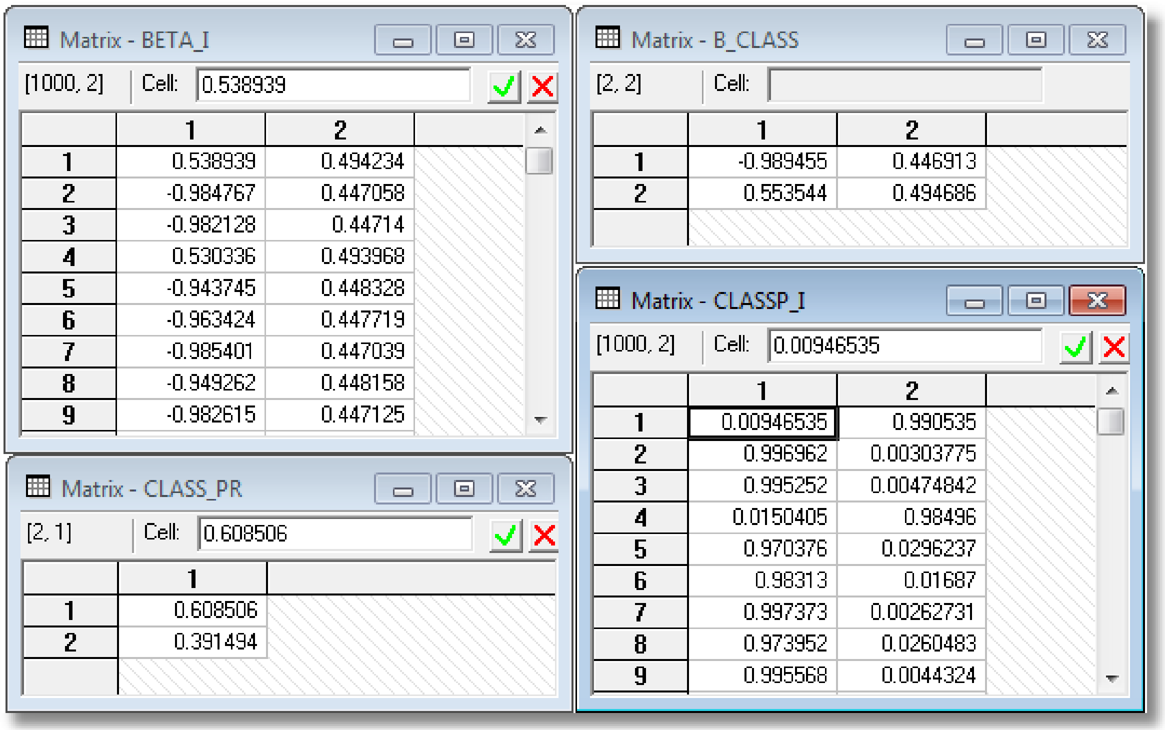
The **; Parameters** specification in the command also creates three matrices:

b_class	= coefficient matrix - one column for each class, coefficients for underlying models,
class_pr	= estimated class probabilities,
beta_i	= group (individual) specific posterior estimated coefficient vector.

```
CALC      ; Ran (123457) $
SAMPLE    ; 1-5000 $
MATRIX    ; ui = Rndm(1000) $
CREATE    ; x = Rnn(0,1) ; i = Trn(5,0) $
CREATE    ; yits = -.5 + .5*x + ui(i) + Rnn(0,1) $
CREATE    ; yit = yits > 0 $
PROBIT    ; Lhs = yit ; Rhs = one,x
          ; Pds = 5
          ; LCM
          ; Pts = 2
          ; Parameters
          ; List $
```

```
-----------------------------------------------------------------------------
Probit   Regression Start Values for YIT
Dependent variable                  YIT
Log likelihood function     -3154.28695
Estimation based on N =   5000, K =   2
Inf.Cr.AIC  = 6312.574 AIC/N =    1.263
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    -.32119***      .01850   -17.37  .0000     -.35744   -.28494
       X|     .34595***      .01912    18.09  .0000      .30847    .38343
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Normal exit:  10 iterations. Status=0, F=    2859.914
-----------------------------------------------------------------------------
Latent Class / Panel Probit   Model
Dependent variable                  YIT
Log likelihood function     -2859.91433
Restricted log likelihood   -3154.28695
Chi squared [   4 d.f.]       588.74525
Significance level               .00000
McFadden Pseudo R-squared      .0933246
Estimation based on N =   5000, K =   5
Inf.Cr.AIC  = 5729.829 AIC/N =    1.146
Sample is  5 pds and   1000 individuals
PROBIT (normal)  probability model
Model fit with  2 latent classes.
```

```
--------+--------------------------------------------------------------------
        |                  Standard            Prob.       95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*          Interval
--------+--------------------------------------------------------------------
        |Model parameters for latent class 1
Constant|    -.98945***       .05685   -17.40  .0000    -1.10089    -.87802
       X|     .44691***       .03553    12.58  .0000      .37728     .51655
        |Model parameters for latent class 2
Constant|     .55354***       .06718     8.24  .0000      .42187     .68522
       X|     .49469***       .03977    12.44  .0000      .41673     .57264
        |Estimated prior probabilities for class membership
Class1Pr|     .60851***       .03119    19.51  .0000      .54738     .66963
Class2Pr|     .39149***       .03119    12.55  .0000      .33037     .45262
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

=============================================================================
Predictions computed for the group with the largest posterior probability
Obs.  Periods Fitted outcomes
=============================================================================
Ind.=    1  J* = 2  P(j)=  .009  .991
       01-05    1.0    1.0    1.0    1.0    1.0
Ind.=    2  J* = 1  P(j)=  .997  .003
       01-05    1.0    1.0    1.0    1.0    1.0
Ind.=    3  J* = 1  P(j)=  .995  .005
       01-05    1.0    1.0    1.0    1.0    1.0
Ind.=    4  J* = 2  P(j)=  .015  .985
```



Matrix - BETA_I
[1000, 2] Cell: 0.538939

	1	2
1	0.538939	0.494234
2	-0.984767	0.447058
3	-0.982128	0.44714
4	0.530336	0.493968
5	-0.943745	0.448328
6	-0.963424	0.447719
7	-0.985401	0.447039
8	-0.949262	0.448158
9	-0.982615	0.447125

Matrix - B_CLASS
[2, 2] Cell:

	1	2
1	-0.989455	0.446913
2	0.553544	0.494686

Matrix - CLASS_PR
[2, 1] Cell: 0.608506

	1
1	0.608506
2	0.391494

Matrix - CLASSP_I
[1000, 2] Cell: 0.00946535

	1	2
1	0.00946535	0.990535
2	0.996962	0.00303775
3	0.995252	0.00474842
4	0.0150405	0.98496
5	0.970376	0.0296237
6	0.98313	0.01687
7	0.997373	0.00262731
8	0.973952	0.0260483
9	0.995568	0.0044324

Figure R22.4 Latent Class Model Results

R23: Fixed and Random Effects Models for Panel Data

R23.1 Introduction

LIMDEP provides a variety of panel data treatments for about 50 different model specifications. The starting point is the basic common effects model, in the form of fixed and random effects. These versions of each model, such as the linear model, Poisson regression, probit model, or ordered choice models, are described in detail in the *Econometric Modeling Guide*. In this chapter, we will detail some general results about how fixed and random effects are handled in linear and nonlinear models.

R23.2 Fixed Effects Models

The fixed effects linear regression model is fit by ordinary least squares, instrumental variables or feasible generalized least squares in the presence of autocorrelation. For other models, fixed effects models are fit by maximum likelihood methods. Fixed effects nonlinear models have heretofore been viewed generally as intractable except in a few special cases. The problem has been that it is generally not possible to transform the data in such a way as to remove the effects from the density, so the researcher was faced with the need to fit all the model coefficients. We have developed a method of handling this problem (in collaboration with George Jakubson of Cornell University) which essentially estimates the model by brute force, but takes advantage of some special characteristics of the fixed effects model. In a few familiar cases (three), it is possible to condition the fixed effects out of the density and estimate the parameters of interest via a conditional log likelihood. For our purposes, this is a partial solution, as it provides a method for some useful cases, logit and Poisson, for example, but does not extend to some very important cases, namely the tobit and probit models. The methods we have developed for *LIMDEP* extend to these three and many other interesting cases.

R23.2.1 Least Squares in the Linear Regression Model

The one way fixed effects linear regression model is described in Chapter E17. The model is

$$y_{it} = \alpha_i + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it}$$

The model is fit by taking advantage of the Frisch-Waugh theorem for partitioned least squares regression. The least squares slope coefficients are obtained by linear regression of

$$(y_{it} - \bar{y}_{i.}) \text{ on } (\mathbf{x}_{it} - \bar{\mathbf{x}}_{i.}) \text{ where } \bar{y}_{i.} = \sum\nolimits_{t=1}^{T_i} y_{it} \text{ and } \bar{\mathbf{x}}_{i.} = \sum\nolimits_{t=1}^{T_i} \mathbf{x}_{it}$$

Panels may be unbalanced, as usual. By this construction, a separate constant term is estimated for each group, as opposed to an overall constant and ‘contrasts’ for *N*-1 of the groups. The individual constant terms are then computed as the group specific residuals,

$$a_i = \sum\nolimits_{t=1}^{T_i} (y_{it} - \bar{y}_{i.}) - (\mathbf{x}_{it} - \bar{\mathbf{x}}_{i.})'\mathbf{b}.$$

When there is autocorrelation, the model is first partial differenced using the value of *rho* provided from your prior estimation of the model – see Chapter E11. The fixed effects treatment is applied to the transformed model,

$$y_{it} - r\,y_{i,t\text{-}1} = \boldsymbol{\beta}'(\mathbf{x}_{it} - r\,\mathbf{x}_{i,t\text{-}1}) + \alpha_i(1 - r) + (\varepsilon_{it} - r\,\varepsilon_{i,t\text{-}1})$$

The fixed effects computed for this model are saved in a matrix named *alphafe*.

The basic command (without the other available options) for this linear regression model is

```
REGRESS      ; Lhs  = the dependent variable
             ; Rhs  = the independent variables, not including one
             ; Pds  = the specification of the periods indicator
             ; Panel
             ; Fixed Effects $ (or just ; Fixed)
```

If you have defined your panel data with **SETPANEL** (see Chapter R5), then the **; Pds = …** may be omitted.

The two way fixed effects linear regression model (see Section E17.3) is

$$y_{it} = \alpha_i + \gamma_t + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it}.$$

It is usually assumed – this is essential for your purposes – that in a two way fixed model, the number of periods in the data set is relatively small – generally less than about 100. Since panels may be unbalanced, there is a large degree of complication in the computation of the estimates. This is handled in *LIMDEP* by treating the two way fixed effects model as a one way model with a set of $T^* - 1 = \text{Max}(T_i) - 1$ period specific dummy variables. Thus, the coefficients in the two way model are actually computed by generating the period specific dummy variables. There is no correction for autocorrelation supported in the two factor fixed framework.

NOTE: The number of periods (or second level groups) may be up to 1,000.

There are counterparts to the simple formulas based on the Frisch-Waugh theorem for two way panel models. In particular, the OLS coefficients can be estimated by least squares regression of the transformed variables. $Dy_{it} = y_{it} - \bar{y}_{i.} - \bar{y}_{.t} + \bar{\bar{y}}_{..}$. However, this only gives the correct answer for a balanced panel. If the panel is unbalanced, this formula will result in a plausible, but incorrect answer for the two way fixed effects model. Because *LIMDEP* never restricts you to a balanced panel for any model, this formula is not used. The two way fixed effects model is always fit by creating the period dummy variables and including them in a one way fixed effects model.

In the reported output for the two way fixed effects model, the fixed effects are transformed so that the model contains an overall constant term and two sets of centered dummy variable coefficients. That is, in the two factor model (not the one factor model), the output will contain an overall constant term and group specific coefficients, a_i transformed so that $\Sigma_i a_i = 0$, and period specific coefficients, c_t transformed so that $\Sigma_t c_t = 0$.

For the linear regression model – this differs a bit from the nonlinear models – you must provide a variable that lists the periods, even if the panel is balanced. Thus, within each group, the variable will take some or all of the values 1,...,T*. For a balanced panel, you can create this with

```
CREATE      ; date = Trn(-t,0) $
```

where T is the number of periods. If the panel is unbalanced, you must create the variable in whatever fashion is necessary. (Some hints appear in Chapter R5.) The model command is then

```
REGRESS     ; Lhs  = the dependent variable
            ; Rhs  = the independent variables, not including one
            ; Pds  = the specification of the periods indicator
            ; Time = the time variable
            ; Fixed Effects $
```

NOTE: In previous versions of *LIMDEP*, the equivalent was **; Period = the time variable**. This is still supported. The form suggested above is provided to maintain consistency with the newer models.

There are other options available for the panel data specifications in the linear model as well. Further details appear in Chapters E16-19. The linear model with fixed effects and endogenous right hand side variables is described in Chapter E22.

Example – Linear Fixed Effects Model

The listing below demonstrates the results for a linear fixed effects model in a (balanced) panel with 595 individuals and seven periods. The commands are

```
SETPANEL    ; Group = _stratum ; Pds = ti $
REGRESS     ; Lhs = lwage ; Rhs = one,wks,occ,ind,south
            ; Panel ; Fixed Effects $
```

The standard results include the simple least squares results without fixed effects followed by the least squares dummy variable (LSDV) estimates. The last table of results presents summary analysis of variance statistics and the results of hypothesis tests of the null model with no effects against the fixed effects model. For these data, the model of the null hypothesis is decisively rejected based on the F statistic and the chi squared results.

```
+----------------------------------------------------------------+
| Variable = _____________ Variable Groups    Max     Min   Average |
| TI         Group sizes  _STRATUM    595      7       7       7.0 |
+----------------------------------------------------------------+

----------------------------------------------------------------------------
Ordinary     least squares regression ............
LHS=LWAGE    Mean                    =       6.67635
             Standard deviation      =        .46151
             No. of observations     =          4165  Degrees of freedom
Regression   Sum of Squares          =       127.319          4
Residual     Sum of Squares          =       759.586       4160
Total        Sum of Squares          =       886.905       4164
             Standard error of e     =        .42731
Fit          R-squared               =        .14355  R-bar squared =   .14273
Model test   F[  4,  4160]           =     174.32156  Prob F > F*   =   .00000
Diagnostic   Log likelihood          =   -2366.09179  Akaike I.C.   = -1.69930
             Restricted (b=0)        =   -2688.80603
             Chi squared [  4]       =     645.42847  Prob C2 > C2* =   .00000
Panel Data Analysis of LWAGE                [ONE way]
               Unconditional ANOVA (No regressors)
Source         Variation   Deg. Free.    Mean Square
Between        646.25374        594.         1.08797
Residual       240.65119       3570.          .06741
Total          886.90494       4164.          .21299
--------+-------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
   LWAGE|  Coefficient       Error       z    |z|>Z*         Interval
--------+-------------------------------------------------------------------
     WKS|     .00518***      .00129     4.01  .0001      .00265    .00772
     OCC|    -.30924***      .01362   -22.70  .0000     -.33595   -.28254
     IND|     .10092***      .01397     7.22  .0000      .07354    .12830
   SOUTH|    -.16271***      .01467   -11.09  .0000     -.19146   -.13397
Constant|    6.59913***      .06126   107.73  .0000     6.47906   6.71919
--------+-------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
----------------------------------------------------------------------------

----------------------------------------------------------------------------
LSDV         least squares with fixed effects ....
LHS=LWAGE    Mean                    =       6.67635
             Standard deviation      =        .46151
             No. of observations     =          4165  Degrees of freedom
Regression   Sum of Squares          =       647.046        598
Residual     Sum of Squares          =       239.859       3566
Total        Sum of Squares          =       886.905       4164
             Standard error of e     =        .25935
Fit          R-squared               =        .72955  R-bar squared =   .68420
Model test   F[598,  3566]           =      16.08638  Prob F > F*   =   .00000
Diagnostic   Log likelihood          =      34.44842  Akaike I.C.   = -2.56678
             Restricted (b=0)        =   -2688.80603
Estd. Autocorrelation of e(i,t)      =       .492839
Panel:Groups Empty      0,     Valid data      595
             Smallest   7,     Largest           7
             Average group size in panel      7.00
Variances    Effects a(i)         Residuals e(i,t)
              .143678                    .067263
```

```
--------+--------------------------------------------------------------------
        |                  Standard             Prob.       95% Confidence
   LWAGE|  Coefficient       Error       z     |z|>Z*          Interval
--------+--------------------------------------------------------------------
     WKS|      .00111         .00102     1.09  .2766     -.00089     .00311
     OCC|    -.07139***       .02331    -3.06  .0022     -.11707    -.02571
     IND|      .02944         .02632     1.12  .2634     -.02215     .08103
   SOUTH|    -.02694          .05818     -.46  .6433     -.14098     .08710
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

+----------------------------------------------------------------------+
|              Test Statistics for the Classical Model                  |
+----------------------------------------------------------------------+
|      Model               Log-Likelihood    Sum of Squares  R-squared |
|(1)  Constant term only    -2688.80597          886.90494     .00000 |
|(2)  Group effects only       27.58464          240.65119     .72866 |
|(3)  X - variables only    -2366.09173          759.58557     .14355 |
|(4)  X and group effects      34.44849          239.85932     .72955 |
+----------------------------------------------------------------------+
|                          Hypothesis Tests                             |
|         Likelihood Ratio Test              F Tests                    |
|         Chi-squared   d.f.   Prob          F    num   denom  P value |
|(2) vs (1)   5432.78    594  .0000      16.14   594    3570   .00000 |
|(3) vs (1)    645.43      4  .0000     174.32     4    4160   .00000 |
|(4) vs (1)   5446.51    598  .0000      16.09   598    3566   .00000 |
|(4) vs (2)     13.73      4  .0082       2.94     4    3566   .01912 |
|(4) vs (3)   4801.08    594  .0000      13.01   594    3566   .00000 |
+----------------------------------------------------------------------+
```

R23.2.2 Maximum Likelihood Estimation

The general (possibly nonlinear) one way fixed effects model is

$$z_{it} = \alpha_i + \boldsymbol{\beta}'\mathbf{x}_{it},\ i = 1,\ldots,N,\ t = 1,\ldots,T_i,$$
$$p(y_{it}) = g(z_{it}, \boldsymbol{\theta})$$

where α_i is the coefficient on a binary variable, d_i, which indicates membership in the ith group. The panel is assumed to consist of N groups with T_i observations in the ith group. The panel need not be balanced; T_i may always vary across groups. Nonlinear models of this form are estimated in two ways. A *conditional* estimator is obtained by using the conditional joint distribution, $f(y_{i1},y_{i2},\ldots,y_{iT}|\Sigma_t y_{it})$. (See, for example Griliches, Hall, and Hausman (1984) who develop this for the Poisson regression.) The resulting density is a function of $\boldsymbol{\beta}$ alone, which is then estimated by (conditional) maximum likelihood. This estimator is available for the binary logit, Poisson, and negative binomial models. Chapters E30 and E44 provide extensive details. Other models do not reduce to a useable closed form through this conditioning, so that the conditional estimator is unavailable. The *unconditional* estimator is obtained by a direct maximization of the full log likelihood function and estimating all parameters including the group specific constants.

Maximum Number of Groups

The estimator described here actually computes the full vector of K+N coefficients, ‘the hard way,’ one might say. This means that your estimated coefficient vector can be huge.

The upper limit on the number of group specific constants that may be estimated is 250,000.

Estimated Coefficients in Fixed Effects Models

As seen in the example below, the model output does not include the estimated fixed effects (dummy variable) coefficients. These are saved in a matrix named *alphafe*, but not displayed. You can examine the matrix by double clicking its name in the project window. The earlier example demonstrates.

Certain models, such as the binary choice models, require that observation groups in which there is no variation in the dependent variable be dropped from the sample when estimating fixed effects models – there is no MLE for the individual specific constant in this case. (This is not general – it only applies to a few models.) In a probit model, for example, the individual specific coefficient cannot be estimated if the Lhs variable is always zero or always one. The *alphafe* matrix’s length matches the number of groups in all cases. Noncomputable αs, e.g. for logit and probit, are set equal to -1.d20 or +1.d20 when y_{it} = 0 or 1, respectively.

The Incidental Parameters Problem

Full estimation of the fixed effects model in this fashion generally encounters the ‘incidental parameters’ problem. The estimators of the fixed effects coefficients are inconsistent in a fixed effects model, not because they estimate the wrong parameters, but because the variances of the estimators of α_i are of order $1/T_i$ which is not assumed to be increasing, not $1/N$, which is. Thus, the properties of the slope estimator (and the estimator of θ in the negative binomial model) depend on an inconsistent estimator. It can be shown (see below) that the variance of the slope estimator converges to zero. The mean of the slope estimator converges to a function that deviates from $\boldsymbol{\beta}$ as a function of the extent to which the estimator of α_i deviates from α_i. Let a_i be the MLE of α_i and $\mathbf{b}$ be the estimator of $\boldsymbol{\beta}$. The usual results for the MLE in a multiparameter situation would produce consistency from the fact that $\mathbf{b} = \mathbf{b}(a_1, a_2, ...)$ and

$$\text{plim}_{N\to\infty}\,\mathbf{b} - \boldsymbol{\beta} \;=\; \text{a function of, among other things, } \text{plim}_{N\to\infty}\, a_i - \alpha_i,\ i = 1,...,N.$$

In the usual case, all terms (including the other estimators) would converge to zero. In this case, that does not hold, though the extent to which the small sample (T_i) affects $\mathbf{b}$ is unknown. (Contrary to widespread belief, the bias of the MLE is not always upward. In the tobit model, there appears to be none, though the estimator of σ is biased downward, while in the truncated regression, it appears that both the slopes and the estimator of σ are biased toward zero. Unfortunately, the end result is strongly model specific. See Greene (2004b). Certainly if your panel contains very small group sizes, say T_i less than five or so, then this estimator is shaky. If you have fairly large group sizes, say on the order of 15 or more, then you are in the range of sample sizes that analysts often rely upon to assert other asymptotic results. Users are urged to consider this issue when using the unconditional fixed estimators.

Surprisingly, the incidental parameters problem is not present in the Poisson model. The reason for this intriguing result is that in the Poisson model, the first order conditions for estimation of the slopes are actually free of the fixed effects – see Winkelmann (2000) for a proof. You can see this effect at work in the application in Chapter E44. The conditional and unconditional estimators are identical. This is not the case for the negative binomial or binary logit models, however. It is for a few other estimators, such as the exponential regression. The formal reason for this result is that in the cases listed, there exist sufficient statistics, usually the group means of the dependent variable, for estimation of the dummy variable coefficients.

Two Way Fixed Effects Models

LIMDEP’s unconditional estimator can also produce a two way fixed effects model,

$$z_{it} = \alpha_i + \delta_t + \boldsymbol{\beta}'\mathbf{x}_{it}.$$

There will now be MaxT_i-1 additional coefficients in the model. You can request this estimator by adding

; Time = ti

where the variable *ti* tells, for each observation, in which period the observation occurred. This variable must take the values 1,2,...,MaxT_i. That is, it must be coded with ‘*t*,’ the index number of the period. A date will not work – it will be flagged as identifying too many coefficients. Do note that observations may be made at different periods in the different groups. For example, if you have a panel with three observations in the first group and seven in the second, the first three observations could have been made at $t = 2$, $t = 4$, and $t = 7$. The routine assumes that MaxT_i is equal to the largest group size in the model. (That way, it is assured that there are no holes in the sequence of observations.) Thus, the largest group in the sample must have this variable coded with the complete set of integers, 1,2,...,*Tmax*.

NOTE: If you have a balanced panel with **; Pds = t** where *t* is a fixed value, then you can specify the time effects with **; Time = one** as there can be no variation in the coding of the period in a balanced panel.

NOTE: Our experience has been that this extension produces considerable instability in the negative binomial, though it works nicely in the Poisson model.

The fixed effects model with time effects is estimated by actually creating the time specific dummy variables. You will see a complete set of time effects in the output. As such, however, if you have a large group size in your panel, this extension may create an extremely large model.

R23.2.3 How it's Done

The unconditional log likelihood is maximized by using Newton's method. The log likelihood is

$$\log L = \sum_{i=1}^{n} \log\left[\prod_{t=1}^{T_i} g(y_{it}, z_{it}, \boldsymbol{\theta})\right]$$

Let p_{it}, y_{it}, $\mathbf{x}_{it}$ and z_{it} denote the components of this function. Then, by simple differentiation, we obtain

$$\frac{\partial \log L}{\partial \boldsymbol{\beta}} = \sum_{i=1}^{N}\sum_{t=1}^{T_i} \frac{\partial \log g(y_{it}, z_{it}, \boldsymbol{\theta})}{\partial z_{it}}\mathbf{x}_{it} = \mathbf{g}_{\beta} = \sum_{i=1}^{N}\sum_{t=1}^{T_i} g_{it}\mathbf{x}_{it}$$

$$\frac{\partial \log L}{\partial \alpha_\iota} = \sum_{t=1}^{T_i} \frac{\partial \log g(y_{it}, z_{it}, \boldsymbol{\theta})}{\partial z_{it}} = g_i = \sum_{t=1}^{T_i} g_{it}$$

$$\frac{\partial^2 \log L}{\partial \boldsymbol{\beta}\partial \boldsymbol{\beta}'} = \sum_{i=1}^{N}\sum_{t=1}^{T_i} \frac{\partial^2 \log g(y_{it}, z_{it}, \boldsymbol{\theta})}{\partial z_{it}^{2}}\mathbf{x}_{it}\mathbf{x}_{it}' = \mathbf{H}_{\beta\beta'}$$

$$\frac{\partial^2 \log L}{\partial \alpha_i^2} = \sum_{t=1}^{T_i} \frac{\partial^2 \log g(y_{it}, z_{it}, \boldsymbol{\theta})}{\partial z_{it}^2} = h_{ii}$$

$$\frac{\partial^2 \log L}{\partial \boldsymbol{\beta}\partial \alpha_i} = \sum_{t=1}^{T_i} \frac{\partial^2 \log g(y_{it}, z_{it}, \boldsymbol{\theta})}{\partial z_{it}^2}\mathrm{x}_{it} = \mathbf{h}_{\beta i}$$

Assemble the full set of first derivatives in a $(K+N)\times 1$ vector, **g** and the full set of second derivatives in a $(K+N)\times(K+N)$ matrix, **H**. The iteration for Newton's method is

$$\begin{aligned}\gamma_{s+1} &= \gamma_s - \mathbf{H}_s^{-1}\mathbf{g}_s \\ &= \gamma_s + \mathbf{d}_s,\end{aligned}$$

where $\boldsymbol{\gamma}$ denotes the full $(K+N)\times 1$ parameter vector, $(\boldsymbol{\beta}', \alpha_1, \alpha_2, \ldots, \alpha_N)'$ and s indexes iterations. This iteration then, computes a change vector, $\mathbf{d}_s$ as the product of the matrix and vector of derivatives. In principle, the matrix **H** is huge, which makes this computation unwieldy. However, the lower right $N\times N$ submatrix of **H** (the very large part) is a diagonal matrix – see above. Therefore, it is not necessary actually to compute the entire matrix. The change vector can be computed as a sum of $K\times 1$ vectors which are themselves functions only of the scalar diagonal parts of the submatrix and the $K\times K$ submatrix at the upper left, all of which is very easily done and requires no more computer memory than a conventional estimator, say least squares for a regression.

We use Newton's method for the computations, so the actual Hessian is available for estimation of the asymptotic covariance matrix of the estimators. Let $\mathbf{H}_{\beta\alpha'}$ denote the $K\times N$ submatrix of $\mathbf{H}$ obtained as $[\mathbf{h}_{\beta 1}, \mathbf{h}_{\beta 2}, ..., \mathbf{h}_{\beta N}]$ and let $\mathbf{H}_{\alpha\alpha'}$ denote the $N\times N$ diagonal lower right submatrix of $\mathbf{H}$ obtained as diag$[h_{ii}]$. Then, the estimator of the asymptotic covariance matrix for the MLE of $\boldsymbol{\beta}$ is the upper left submatrix of $-\mathbf{H}^{-1}$. Using the partitioned inverse formula, this is

$$\text{Asy.Var}[\mathbf{b}] = [\mathbf{-(H_{\beta\beta'}} - \mathbf{H}_{\beta\alpha'}(\mathbf{H}_{\alpha\alpha'})^{-1}\,\mathbf{H}_{\alpha\beta'})]^{-1}$$

The first matrix is given above. By inserting the formulas given above, and exploiting the fact that $\mathbf{H}_{\alpha\alpha'}$ is a diagonal matrix, we obtain the simple result

$$\mathbf{H}_{\beta\alpha'}(\mathbf{H}_{\alpha\alpha'})^{-1}\,\mathbf{H}_{\alpha\beta'} = \sum_{i=1}^{N} \frac{1}{h_{ii}}(\mathbf{h}_{\beta i})(\mathbf{h}_{\beta i})'$$

This produces a sum of $K\times K$ matrices which is of the form of a moment matrix and which is easily computed. Thus, the asymptotic covariance matrix for the estimated coefficient vector is easily obtained in spite of the size of the problem.

Two considerations remain. First, it is not possible to store the asymptotic covariance matrix for the estimator of the fixed effects (unless there are relatively few of them). Using the partitioned inverse formula once again, we can show that the elements of Asy.Var[**a**] are contained in

$$\text{Asy.Var}[\mathbf{a}] = [-(\mathbf{H}_{\alpha\alpha'} - \mathbf{H}_{\alpha\beta'}(\mathbf{H}_{\beta\beta'})^{-1}\mathbf{H}_{\beta\alpha'})]^{-1}.$$

The *ij*th element of the matrix to be inverted is

$$(\mathbf{H}_{\alpha\alpha'} - \mathbf{H}_{\alpha\beta'}(\mathbf{H}_{\beta\beta'})^{-1}\mathbf{H}_{\beta\alpha'})_{ij} = \mathbf{1}(i=j)h_{ii} - \mathbf{h}_{\beta i}'(\mathbf{H}_{\beta\beta'})^{-1}\,\mathbf{h}_{\beta j}$$

This is a full $N\times N$ matrix, and so the model size problem will apply – it is not feasible to manipulate this matrix. On the other hand, one could extract particular parts of it if that were necessary. For the interested practitioner, we will lay out the computational results. The Hessian to be inverted for the asymptotic covariance matrix of **a** is

$$\mathbf{H}_{\alpha\alpha'} - \mathbf{H}_{\alpha\beta'}(\mathbf{H}_{\beta\beta'})^{-1}\mathbf{H}_{\beta\alpha'}$$

We keep in mind that $\mathbf{H}_{\alpha\alpha'}$ is an $N\times N$ diagonal matrix. Using result A-66b in Greene (2012), we have that the inverse of this matrix is

$$[\mathbf{H}_{\alpha\alpha'}]^{-1} + [\mathbf{H}_{\alpha\alpha'}]^{-1}\mathbf{H}_{\alpha\beta'}\{(\mathbf{H}_{\beta\beta'})^{-1} - \mathbf{H}_{\beta\alpha'}[\mathbf{H}_{\alpha\alpha'}]^{-1}\,\mathbf{H}_{\alpha\beta'}\}^{-1}\,\mathbf{H}_{\beta\alpha'}[\mathbf{H}_{\alpha\alpha'}]^{-1}.$$

This is a messy expression, but the fact that $\mathbf{H}_{\alpha\alpha'}$is diagonal simplifies it considerably. In particular, by expanding the summations where needed, we find

$$\text{-}Asy.Cov\left[a_i, a_j\right] = 1(i=j)\frac{1}{h_{ij}} + \frac{1}{h_{ii}}\frac{1}{h_{jj}}\mathbf{h}_{\beta i}'\left[\mathbf{H}_{\beta\beta'}^{-1} - \sum_{i=1}^{N}\frac{1}{h_{ii}}\mathbf{h}_{\beta i}\mathbf{h}_{\beta j}'\right]^{-1}\mathbf{h}_{\beta\varphi}$$

(Note that the first term is a bit ambiguously defined, as if $i \neq j$, this is 0/0 – we define it to be zero.) At first glance, this may appear not to have gained anything. But, it has gained a great deal. The matrix to be inverted is $K\times K$, not $N\times N$, so this can be computed by summation. It may be a large amount of computing, but it is at least straightforward, and easily calculated as it involves only $K\times 1$ vectors and repeated use of the same $K\times K$ inverse matrix. Note that this is the inverse of the Hessian, which must be inverted to compute the asymptotic variance or covariance. Hence the leading minus sign at the left of the definition.

Likewise, the asymptotic covariance matrix of the slopes and the constant terms can be arranged in a computationally feasible format. Using what we already have and result A-74 in Greene (2012), we find that

$$Asy.Cov[\mathbf{b},\mathbf{a}'] = -(\mathbf{H}_{\beta\beta'})^{-1}\,\mathbf{H}_{\beta\alpha'}\times \text{Asy.Var}[\mathbf{a}]$$

Once again, this involves $N\times N$ matrices, but the expression simplifies. Using our previous results, we can reduce this to

$$Asy.Cov[\mathbf{b},a_i] = -(\mathbf{H}_{\beta\beta'})^{-1}\sum_{m=1}^{N}\mathbf{h}_{\beta i}\,Asy.Cov[a_i,a_m]\,.$$

Again, the gain in simplification may not be obvious, but it is substantial. This asymptotic covariance matrix involves a huge amount of computation, but essentially no computer memory – only the $K\times K$ matrix. The $K\times 1$ vectors would have to be computed ‘in process,’ which is why this involves a large amount of computation. But, once again, it is very feasible. At no point is it necessary to maintain an $N\times N$ matrix, which has always been viewed as the obstacle. Finally, we note the motivation for the last two results. One might be interested in the computation of an asymptotic variance for a function $g(\mathbf{b},a_i)$ such as a prediction for a probit model, $\Phi(\mathbf{b}'\mathbf{x}_{it} + a_i)$. The delta method would require a very large amount of computation, but it is feasible with the preceding results.

Finally, note that in Asy.Var[**b**], the terms are of order (NT) minus a sum of N order T outer products. Therefore, the end result is the inverse of an order NT matrix, which will converge to zero. What this establishes is that **b** does converge to a parameter in the sense that its asymptotic covariance matrix converges to zero. However, it converges to a function that deviates from $\boldsymbol{\beta}$ to the extent that plim a_i deviates from α_i.

Example – Nonlinear Fixed Effects Model

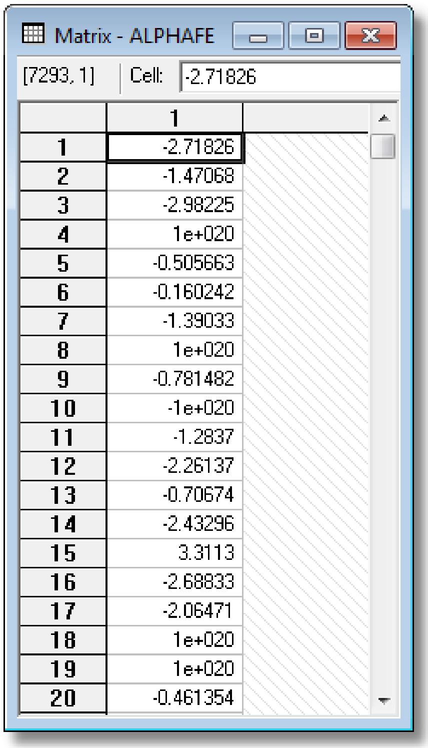
The listing below shows a fixed effects model for an unbalanced with 7,293 groups.

```
SAMPLE      ; All $
SETPANEL    ; Group = id ; Pds = ti $
PROBIT      ; Lhs = doctor ; Rhs = age,educ,hhninc,married
            ; FEM ; Panel ; Par
            ; Partial Effects $
```

```
+-------------------------------------------------------------------+
| Variable = ____________ Variable Groups    Max    Min    Average |
| TI          Group sizes  ID          7293      7      1        3.7 |
+-------------------------------------------------------------------+

-----------------------------------------------------------------------------
Probit   Regression Start Values for DOCTOR
Dependent variable               DOCTOR
Log likelihood function    -17700.96342
Estimation based on N =  27326, K =   5
Inf.Cr.AIC  =35411.927 AIC/N =    1.296
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     AGE|     .01539***      .00072    21.42  .0000      .01398    .01679
    EDUC|    -.02811***      .00350    -8.03  .0000     -.03497   -.02125
  HHNINC|    -.09776**       .04626    -2.11  .0346     -.18844   -.00708
 MARRIED|    -.00931         .01888     -.49  .6220     -.04630    .02769
Constant|     .02642         .05397      .49  .6244     -.07936    .13221
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

Nonlinear Estimation of Model Parameters
Method=Newton; Maximum iterations=100
Convergence criteria: max|dB|   .1000D-05, dF/F=  .1000D-08, g<H>g=  .1000D-08
Normal exit from iterations. Exit status=0.

-----------------------------------------------------------------------------
FIXED EFFECTS Probit Model
Dependent variable               DOCTOR
Log likelihood function     -9454.05945
Estimation based on N =  27326, K =4251
Inf.Cr.AIC  =27410.119 AIC/N =    1.003
Unbalanced panel has   7293 individuals
Skipped 3046 groups with inestimable ai
PROBIT (normal)  probability model
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
     AGE|     .06334***      .00426    14.87  .0000      .05499    .07169
    EDUC|    -.07547*        .04063    -1.86  .0632     -.15510    .00416
  HHNINC|    -.02496         .10713     -.23  .8158     -.23493    .18501
 MARRIED|    -.04865         .06194     -.79  .4322     -.17004    .07275
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
-----------------------------------------------------------------------------
Partial derivatives of E[y] = F[*]   with
respect to the vector of characteristics.
They are computed at the means of the Xs.
Estimated E[y|means,mean alphai]=    .621
Estimated scale factor for dE/dx=    .380
--------+--------------------------------------------------------------------
        |     Partial                          Prob.      95% Confidence
  DOCTOR|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     AGE|     .02409***      1.66920     6.94  .0000      .01729     .03089
    EDUC|    -.02871         -.52464    -1.48  .1392    -.06675     .00934
  HHNINC|    -.00949         -.00540     -.23  .8151    -.08906     .07007
 MARRIED|    -.01850***      -.02978    -3.34  .0008    -.02935    -.00766   #
--------+--------------------------------------------------------------------
#  Partial effect for dummy variable is E[y|x,d=1] - E[y|x,d=0]
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The vector *alphafe* contains the estimated fixed effects coefficients for those groups for which it is estimable. The vector shows (rows 4, 8, 18, 19) observations at which there is no within group variation in the dependent variable.



	1
1	-2.71826
2	-1.47068
3	-2.98225
4	1e+020
5	-0.505663
6	-0.160242
7	-1.39033
8	1e+020
9	-0.781482
10	-1e+020
11	-1.2837
12	-2.26137
13	-0.70674
14	-2.43296
15	3.3113
16	-2.68833
17	-2.06471
18	1e+020
19	1e+020
20	-0.461354

Figure R23.1 Estimated Fixed Effects

R23.3 Random Effects Models

The random effects model is formulated in the context of an index function model,

$$P(y_{it} \mid \mathbf{x}_{it}, u_i) \;=\; g(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} \;+\; u_i, \boldsymbol{\theta})$$

where $\boldsymbol{\theta}$ is any ancillary parameters that appear in the model and u_i, which is time invariant, is the latent unobserved heterogeneity that enters the model in the form of the random effect. The model command will typically be of the form

```
Model name such as PROBIT, LOGIT, etc.
                    ; Lhs  = ...
                    ; Rhs  = ...
                    ; Pds  = panel specification
                    ; Random Effects  $
```

(There is some variation in the commands for specific models.) Other options follow the usual pattern, and are described for each model in the chapters to follow.

All models that support panel data treatments in *LIMDEP* provide some kind of random effects estimator. In many cases, more than one is available. In addition, various estimation techniques are used in *LIMDEP* to fit random effects models. These are summarized in Table R23.1 below. The various estimation techniques are described at several points in this manual. This section will collect a few common results. Some general observations:

- The two step FGLS procedure is specific to the linear regression model, and is described in detail in Chapter E18 with other methods for estimation of the linear model with panel data.

- Four models, Poisson, negative binomial, the parametric survival models with heterogeneity and several forms of the stochastic frontier models have known closed forms for the unconditional distribution of the observed response – that is, the density after u_i is integrated out. These are described in detail in the specific chapters devoted to these models.

- The random parameters model is described in lengthy detail in Chapter R24. As noted, in any model included in that framework, a pure random effects model results if the constant term is treated as the only random parameter. In these cases, the simulation estimator provides an alternative to the quadrature based estimator that we describe here.

- The linear regression model with random coefficients can be estimated in two ways. A modified generalized least squares procedure due to Hildreth, Houck, and Swamy is presented in Section E15.4. Through appropriate restrictions on the model, this can be forced to be a random effects model. However, this is the third best of three approaches to random effects in the linear model. (We mention it purely for completeness.)

Model Class	Two Step FGLS	Random Constant in RPM	ML with Quadrature	Exact ML
Linear Regr. 1 & 2 Way	●	●		●
Linear Reg. Multilevel		●		
Binary Choice				
Probit		●	●	
Logit		●	●	
Complementary Log Log		●	●	
Gompertz		●	●	
Bivariate Probit		●		
Bivar. Probit Selection		●		
Partial Observability		●		
Multinomial Choice				
Multinomial Logit		●		
Multinomial Probit		●		
Ordered Probability/All		●	●	
Generalized Ord. Probit		●		
Count Data				
Poisson Regression		●	(some forms)	●
Negative Binomial		●		●
Poisson/NegBin ZIP		●		
Loglinear Models				
Exponential		●		
Gamma		●		
Weibull		●		
Inverse Gaussian		●		
Geometric		●		
Power		●		
Binomial		●		
Normal (Exp Regression)		●		
Limited Dependent Variable				
Tobit		●	●	
Censored (Grouped) Data		●		
Truncated Regression		●		
Sample Selection		●		
Survival and Frontier Models				
Weibull		●		●
Exponential		●		●
Loglogistic		●		●
Lognormal		●		●
Stochastic Frontier		●		●

Table R23.1 Random Effects Model Estimators

There are additional forms of the random effects model. The linear regression model supports a three level nested random effect model,

$$y_{ijkt} = \boldsymbol{\beta}'\mathbf{x}_{ijkt} + w_{ijk} + v_{ij} + u_i + \varepsilon_{ijkt},$$

which is estimated by maximum likelihood (assuming all components are normally distributed). This estimator is described in Section E18.8. The Hausman and Taylor model modifies the one way random effects model to accommodate correlation among some of the right hand side variables and the random effect. The formulation is

$$y_{it} = \boldsymbol{\beta}_1'\mathbf{x}_{1,it} + \boldsymbol{\beta}_2'\mathbf{x}_{2,it} + \boldsymbol{\gamma}_1'\mathbf{f}_{1,i} + \boldsymbol{\gamma}_2'\mathbf{f}_{2,i} + u_i + \varepsilon_{it},$$

where $\mathbf{x}_{2,it}$ and $\mathbf{f}_{2,i}$ are correlated with u_i. Discussion appears in Chapter E23. The Arellano, Bond and Bover estimator for dynamic panel data models extends this formulation to add a lagged dependent variable,

$$y_{it} = y_{i,t-1} + \boldsymbol{\beta}_1'\mathbf{x}_{1,it} + \boldsymbol{\beta}_2'\mathbf{x}_{2,it} + \boldsymbol{\gamma}_1'\mathbf{f}_{1,i} + \boldsymbol{\gamma}_2'\mathbf{f}_{2,i} + u_i + \varepsilon_{it},$$

Any random parameters model that uses the methods described below in this section may also include multilevel (up to 10 levels) random effects, with main linear effects and products. The extension of the model takes the form

$$Index_{it} = \boldsymbol{\beta}_i'\mathbf{x}_{\mathrm{it}} + c_{j1}\, e_{j1} + c_{j2}\, e_{j2} + \ldots + c_{jM}\, e_{jM}$$

with up to 10 effects in total. The c_{jm} are ones and zeros simply used to select the effects in the model. The effects are up to 10 normally distributed random terms associated with discrete indicators. Effects may appear singly or as products, and may be nested or simply be associated with any desired groupings of the data. Full details appear in Section R24.8.

R23.3.1 Quadrature Based Estimation – The Butler and Moffitt Method

Write the one way random effects model as

$$z_{it} \mid u_i = \boldsymbol{\beta}'\mathbf{x}_{it} + \sigma_u u_i$$

where $u_i \sim N[0,1]$, and let ε_{it} be the stochastic term in the model that provides the *conditional* distribution. Thus,

$$\mathrm{P}[y_{it} \mid \mathbf{x}_{it}, u_i] = g(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \sigma u_i, \theta),\ i = 1,\ldots,N,\ t = 1,\ldots,T_i.$$

where $g(.)$ is the density discussed earlier (Poisson, normal, logistic, extreme value, Gompertz, etc.). The parameter vector for the random effects model is

$$\boldsymbol{\theta} = [\beta_1,\ldots,\beta_K, \sigma]'.$$

The log likelihood function is

$$\log L = \Sigma_i \log L_i$$

where log L_i is the contribution of the *i*th individual (group) to the total. Conditioned on u_i, the T_i terms in the contribution to the likelihood for group *i* are independent. So, the joint conditional probability for the *i*th group is

$$\mathrm{P}[y_{i1},...,y_{iTi} \mid \mathbf{x}_{i1,...,}u_i] = \prod_{t=1}^{T_i} g(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \sigma_u u_i)$$

where now, u_i is normalized to unit variance. Since u_i is unobserved, it is necessary to obtain the unconditional log likelihood by taking the expectation of this over the distribution of u_i. For convenience, write the *t*th term in the probability above as

$$G(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i),$$

where $\gamma = \sigma_u$, so that

$$L_i | u_i = \prod_{t=1}^{T_i} G(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i).$$

Then,

$$L_i = \mathrm{E}_{ui}[L_i \mid u_i]$$

$$= \int_{-\infty}^{\infty} \frac{\exp(-u_i^2/2)}{\sqrt{2\pi}} \prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i)\, du_i$$

NOTE: It can be seen in the likelihood function that it is necessary actually to compute the product of the densities for the group, not the sum of the logs. For this reason, the number of observations in a group cannot be extremely large. (We are frequently asked about this.) Since the individual density is likely to be on the order of .25 or so, the product of 100 probabilities is on the order of 10^{-100}. This means that the end result is more rounding error than result. In worse cases, the computation will 'overflow' – that is, exceed the computer's capacity to compute the value. For example, the correct result for the product of 100 probabilities on the order of .01 cannot be computed in the accuracy of the computer, which is about $10^{+/-380}$. The diagnostic that this estimator produces mentions a 'Bad counter...' When the counter for group size exceeds 100, the estimator assumes that you have made some kind of error.

Finally,

$$\log L = \sum_{i=1}^{N} \log L_i$$

The function is maximized by solving the likelihood equations:

$$\frac{\partial \log L}{\partial \begin{pmatrix} \boldsymbol{\beta} \\ \gamma \end{pmatrix}} = \sum_{i=1}^{N} \frac{\partial \log L_i}{\partial \begin{pmatrix} \boldsymbol{\beta} \\ \gamma \end{pmatrix}} = \mathbf{0}.$$

For convenience below, let $\boldsymbol{\theta}$ denote the full parameter vector, $[\boldsymbol{\beta},\gamma]'$.

The integration is done with Hermite quadrature. Make the change of variable to $v_i = u_i/\sqrt{2}$. Then,

$$\log L_i \;=\; \log \frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}\exp(-v_i^2)\prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta v_i)\, dv_i$$

where $\delta = \gamma \times \sqrt{2}$. The integral of the form

$$\int_{-\infty}^{\infty}\exp(-v^2)g(v)dv$$

is approximated by the Hermite quadrature,

$$\int_{-\infty}^{\infty}\exp(-v^2)g(v)dv \approx \sum\nolimits_{h=1}^{H} w_h g(z_h)$$

where w_h are the weights and z_h are the abscissas for the approximation. (See Section R23.3.1, Butler and Moffitt (1982) and Abramovitz and Stegun (1972) for further details.) Collecting terms, then, the log likelihood is computed with

$$\log L \;=\; \sum\nolimits_{i=1}^{N} \;\log L_i \approx \sum\nolimits_{i=1}^{N} \;\log\left\{\frac{1}{\sqrt{\pi}}\sum\nolimits_{h=1}^{H} w_h\left[\prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta z_h)\right]\right\}$$

The derivatives of the log likelihood function are approximated as well

$$\frac{\partial \log L_i}{\partial \boldsymbol{\theta}} \;=\; \frac{1}{L_i}\frac{\partial L_i}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial L_i}{\partial \boldsymbol{\theta}} \;=\; \frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty}\exp(-v_i^2)\frac{\partial}{\partial \boldsymbol{\theta}}\prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta v_i)\, dv_i$$

$$\frac{\partial}{\partial \boldsymbol{\theta}}\prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta v_i) \;=\; \sum\nolimits_{t=1}^{T_i}\left[\frac{\partial P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta v_i)}{\partial \boldsymbol{\theta}}\right]\prod_{s\neq t} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{is} + \delta v_i)$$

$$= \prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{is} + \delta v_i)\sum\nolimits_{t=1}^{T_i}\left[\frac{\partial \log P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta v_i)}{\partial \boldsymbol{\theta}}\right]$$

Collecting terms once again, we obtain the approximation,

$$\frac{\partial \log L}{\partial \boldsymbol{\theta}} \;=\; \sum\nolimits_{i=1}^{N} \;\frac{1}{L_i}\frac{\partial L_i}{\partial \boldsymbol{\theta}}$$

$$\approx \sum\nolimits_{i=1}^{N} \frac{\left\{\frac{1}{\sqrt{\pi}}\sum\nolimits_{h=1}^{H} w_h\left[\prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta z_h)\right]\left[\sum\nolimits_{t=1}^{T_i}\frac{\partial \log P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta z_h)}{\partial \boldsymbol{\theta}}\right]\right\}}{\left\{\frac{1}{\sqrt{\pi}}\sum\nolimits_{h=1}^{H} w_h\left[\prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta z_h)\right]\right\}}$$

Note that L_i and its derivatives are approximated separately. The summation involves two separate integrals. We use a 20 point quadrature by default, but you can change the number of quadrature points by including **; Hpt = p** in the command, where '*p*' is the desired number of points, (one of 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 16, 20, 32, 64, 96). In most cases, the accuracy of the computations will improve with the number of quadrature points. However, the amount of computation will increase as well (linearly). The asymptotic covariance matrix estimator is based on the first derivatives, using the BHHH estimator.

As noted, this procedure is used in several models, including the single index binary choice models, the count data models, and the tobit and ordered probability models. The various derivatives and underlying transformations of the parameters will differ a bit from model to model. These are discussed in the specific contexts below. For illustration, consider the common binary choice models, probit and logit. These are single index models that involve only a slope vector, $\boldsymbol{\beta}$. The heterogeneity adds a variance parameter to the model, but the variance term, δ, appears linearly in the function along with $\boldsymbol{\beta}$, so no complication is added by this additional parameter as the summation is done over the abscissas. In each case, the term

$$P(y_{it}, \boldsymbol{\beta}'\mathrm{x}_{\mathrm{it}} + \gamma z_h) \; = \; F\left[\boldsymbol{\beta}'\mathbf{x}_{it} + \gamma z_h\right]^{y_{it}} \left(1 - F\left[\boldsymbol{\beta}'\mathbf{x}_{it} + \gamma z_h\right]\right)^{1-y_{it}}$$

so $$\log P(y_{it}, \boldsymbol{\beta}'\mathrm{x}_{\mathrm{it}} + \gamma z_h) \; = \; y_{it}\log F_{it} \; + \; (1 - y_{it})\log(1 - F_{it}).$$

Thus, $$\frac{\partial \log P(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta z_h)}{\partial \boldsymbol{\theta}} = \left(\frac{y_{it}}{F_{it}} - \frac{1-y_{it}}{1-F_{it}}\right) g_{it}(.) \begin{bmatrix} \mathbf{x}_{it} \\ z_h \end{bmatrix}$$

The functional forms appear in Section E27.2.1. Using the functions defined there, the log derivatives, $g(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i)$ are as follows:

$$\text{Probit:} \;\; g(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i) = \frac{(2y_{it} - 1)\phi(\boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i)}{\Phi[(2y_{it} - 1)(\boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i)]}$$

$$\text{Logit:} \;\; g(y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i) = (2y_{it} - 1)\{1 \; - \; \Lambda[(2y_{it} - 1)(\boldsymbol{\beta}'\mathbf{x}_{it} + \gamma u_i)]\}$$

The asymptotic covariance matrix is estimated by the BHHH estimator,

$$\mathbf{H} \; = \left[\sum_{i=1}^{N} \left(\frac{1}{L_i}\frac{\partial L_i}{\partial \boldsymbol{\theta}}\right)\left(\frac{1}{L_i}\frac{\partial L_i}{\partial \boldsymbol{\theta}}\right)'\right]^{-1}$$

Example – Nonlinear Random Effects Model

An example of the random effects nonlinear model is shown below. The data used are the same as in the fixed effects model estimated earlier.

```
SAMPLE      ; All $
SETPANEL    ; Group = id ; Pds = ti $
PROBIT      ; Lhs = doctor ; Rhs = one,age,educ,hhninc,married
            ; Random Effects ; Panel ; Par
            ; Partial Effects $
```

```
+-------------------------------------------------------------------+
| Variable = ____________ Variable Groups    Max    Min    Average |
| TI          Group sizes  ID          7293      7      1        3.7 |
+-------------------------------------------------------------------+
Normal exit:   4 iterations. Status=0, F=    17701.08

(Same as for the fixed effects model)

Normal exit:  11 iterations. Status=0, F=    16289.92

-----------------------------------------------------------------------------
Random Effects Binary Probit Model
Dependent variable               DOCTOR
Log likelihood function    -16289.42796
Restricted log likelihood  -17700.96342
Chi squared [   1 d.f.]      2823.07092
Significance level               .00000
McFadden Pseudo R-squared      .0797434
Estimation based on N =  27326, K =   6
Inf.Cr.AIC  =32590.856 AIC/N =    1.193
--------+--------------------------------------------------------------------
        |                Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    -.09497        .09406    -1.01  .3127     -.27932    .08938
     AGE|     .02270***     .00125    18.19  .0000      .02025    .02515
    EDUC|    -.03383***     .00629    -5.38  .0000     -.04616   -.02149
  HHNINC|     .02166        .06651      .33  .7447     -.10869    .15201
 MARRIED|    -.04914*       .02934    -1.67  .0940     -.10665    .00838
     Rho|     .45018***     .01020    44.13  .0000      .43019    .47018
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Partial derivatives of E[y] = F[*]  with
respect to the vector of characteristics
They are computed at the means of the Xs
Observations used for means are All Obs.
--------+--------------------------------------------------------------------
        |     Partial                        Prob.      95% Confidence
  DOCTOR|      Effect    Elasticity      z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
     AGE|     .00630***      .42914    18.41  .0000      .00563    .00697
    EDUC|    -.00939***     -.16632    -5.38  .0000     -.01281   -.00597
  HHNINC|     .00601         .00331      .33  .7447     -.03017    .04220
 MARRIED|    -.01364*       -.01619    -1.67  .0941     -.02961    .00233
--------+--------------------------------------------------------------------
z, prob values and confidence intervals are given for the partial effect
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

R24: Random Parameter Models

R24.1 Random Parameters Models

The underlying motivation of the random parameters (RP) model is individual heterogeneity in the parameters in a parametric model. We model this generically as

$$f(y_{it} \mid \mathbf{x}_{it}, \mathbf{z}_i) = g(y_{it}, \mathbf{x}_{it}, \mathbf{z}_i, \boldsymbol{\alpha}_i)$$

where $g(.)$ is the probability density for the observed response of the ith individual at time t, y_{it} is the observed response, $\mathbf{x}_{it}$ and $\mathbf{z}_i$ are measured covariates, and $\boldsymbol{\alpha}_i$ is a person specific parameter vector that varies randomly across individuals, with a mean $\boldsymbol{\alpha}$ and covariance matrix $\boldsymbol{\Omega}$. (The *LIMDEP* implementation does not accommodate heteroscedasticity in the distribution. Heteroscedasticity is supported in the counterpart model for the multinomial logit framework in *NLOGIT* Version 6. Nearly every model supported in *LIMDEP* is included in this framework. The following broad modeling frameworks support this structure:

- Linear regression
- Binary choice: probit, logit, complementary log log, Gompertz, bivariate probit models
- Ordered probability models, probit, logit, Gompertz, complementary log log
- Count data: Poisson and negative binomial, ZIP model and several others
- Censored dependent variable: tobit, grouped data
- Truncated dependent variable: truncated regression
- Loglinear dependent variable: exponential, gamma, inverse Gaussian regression, power, binomial, normal-exponential, beta
- Parametric survival models: Weibull, exponential, lognormal, loglogistic, inverse Gauss
- Stochastic frontier models
- Sample selection models
- Discrete choice: multinomial logit

All of the models listed except the last use the same estimation program and thus have precisely the same structure. The random parameters logit model in *NLOGIT* Version 6 has a separate program for estimation that operates quite similarly, but includes several additional features not contained in the others.

This section describes a broad class of models in *LIMDEP*. This model framework is widely used in many fields of statistics and econometrics. The models described here are also found in other literatures under the headings multilevel models, mixed models, and hierarchical models. All of these have large intersections – some are completely subsumed – in the set of random parameters models described here.

R24.2 Mathematical Formulation of the RP Model

The structure of the random parameters model from the point of view of the modeler is

$\boldsymbol{\alpha}_i \;=\; [\boldsymbol{\beta}_{1i}', \boldsymbol{\beta}_{2i}', \boldsymbol{\theta}']'$

where $\boldsymbol{\theta}$ = ancillary parameters, such as the dispersion parameter in the negative binomial model or σ in a tobit or linear regression model

$\boldsymbol{\beta}_{1i}$ = $\boldsymbol{\beta}_1$ = K_1 nonrandom parameters

$\mathbf{x}_{1it}$ = variables multiplied by $\boldsymbol{\beta}_{1i}$

$\boldsymbol{\beta}_{2i}$ = $\boldsymbol{\beta}_2^0 + \boldsymbol{\Delta}\mathbf{z}_i + \boldsymbol{\Gamma}\mathbf{v}_i$ = K_2 random parameters

where $\boldsymbol{\beta}_2^0$ = the fixed constant terms in the means of the distributions for the random parameters

$\mathbf{z}_i$ = a set of M observed variables which do not vary over time and which enter the means (optional) of the random parameters

$\boldsymbol{\Delta}$ = coefficient matrix, $K_2{\times}M$, which forms the observation specific term in the mean

$\mathbf{v}_i$ = unobservable $K_2{\times}1$ latent random term in the *i*th observation in $\boldsymbol{\beta}_{2i}$. Each element of $\mathbf{v}_{it}$ has zero mean and known variance. Elements of $\mathbf{v}_{it}$ may be distributed as normal, uniform, triangular, lognormal, or others. There are numerous options for specifying the means of the distributions. The distributions of the random parameters need not be the same. Two models are used for the elements of $\mathbf{v}_i$:

Random Effects: $\mathbf{v}_i = \mathbf{v}_i$ for all t. This is the usual random effects form.
Autocorrelated (AR(1)): $\mathbf{v}_{it} = \mathbf{R}\mathbf{v}_{i,t-1} + \mathbf{u}_{it}$ where $\mathbf{R}$ is a diagonal matrix of coefficient specific autocorrelation coefficients and $\mathbf{u}_{it}$ satisfies the earlier specification for $\mathbf{v}_{it}$.

$\boldsymbol{\Gamma}$ = lower triangular or diagonal matrix which produces the covariance matrix of the random parameters, $\boldsymbol{\Omega} = \boldsymbol{\Gamma}\mathbf{A}\boldsymbol{\Gamma}'$ in the random effects form and $\boldsymbol{\Omega} = \boldsymbol{\Gamma}\mathbf{A}^{1/2}(\mathbf{I}\text{-}\mathbf{R}^2)^{-1}\mathbf{A}^{1/2}\boldsymbol{\Gamma}'$ in the AR(1) model. $\mathbf{A}$ is the diagonal matrix of known variances of the elements of $\mathbf{v}_i$. If all parameters are (standard) normally distributed, then $\mathbf{A} = \mathbf{I}$. Uniformly distributed random variables have variance 1/12. Other forms have different values. In the final specification of the model, these implicit scales are absorbed into $\boldsymbol{\Gamma}$ – they will be invisible to you in your estimated model.

$\mathbf{x}_{2it}$ = variables multiplied by $\boldsymbol{\beta}_{2it}$

$\boldsymbol{\beta}_{it}$ = $[\boldsymbol{\beta}_1', \boldsymbol{\beta}_{2it}']'$

With autocorrelated $\mathbf{v}_{it}$, $\boldsymbol{\beta}_i$ can vary with t. For ease of exposition, we will suppress this in what follows, and use $\boldsymbol{\beta}_i$ to denote the random parameter vector. Finally,

$$\mathbf{x}_{it} = [\mathbf{x}_{1it}', \mathbf{x}_{2it}']'$$

$$a_{it} = \boldsymbol{\beta}_{it}'\mathbf{x}_{it}$$

$P(y_i|\mathbf{x}_{it}, \mathbf{z}_i, \mathbf{v}_{it}) = g(y_{it}, a_{it}, \boldsymbol{\theta})$ = the density for the observed response variable.

R24.3 Commands for Random Parameters Models

NOTE: There is no command builder for the random parameters models.

The essential command for the random parameters model is structured as follows, where all parts are mandatory:

Model command such as PROBIT, POISSON, TOBIT, etc.
; Lhs = dependent variable
; Rhs = all variables in x*i*,
including one if model contains a constant
; RPM (for random parameters model)
; Fcn = specification of random parameters $

Panel Data

A panel is specified as usual;

; Pds = specification of number of periods for the panel

The RPM is not strictly for panel data. In principle, the random elements in the parameters serve as the random effects in a panel data model. But, this model can be fit, possibly with less precise results, using a cross section.

Heterogeneity in the Means of the Parameters

As formulated above, the random parameters each have a fixed mean, β_k that is estimated. The general form of the random parameter thus far is

$$\beta_{k,i} = \beta_k^0 + \gamma_k v_{k,i}$$

where β_k is the mean (to be estimated), γ_k is the scale factor and $v_{k,i}$ is a random variable (defined below) The mean may be specified to depend on observed variables $\mathbf{z}_i$ with

; RPM = list of variables in z*i*

The random parameter is now

$$\beta_{k,i} = \beta_k^0 + \boldsymbol{\delta}_k'\mathbf{z}_i + \gamma_k v_{k,i}.$$

The mean vector for the set of random parameters will now be $E[\boldsymbol{\beta}_i|\mathbf{z}_i] = \boldsymbol{\beta}^0 + \boldsymbol{\Delta}\mathbf{z}_i$.

Distributions of Parameters

The **; Fcn** list consists of a list of names of variables that appear in $\mathbf{x}_{2i}$, each followed in parentheses by one of the following distribution specifications: The form is **; Fcn = name(dist)** where *dist* is one of the following:

c	for constant (zero variance), $v_i = 0$
n	for normally distributed, v_i = a standard normally distributed variable
u	for uniform, v_i = a standard uniform distributed variable in (-1,+1)
t	for triangular (the 'tent' distribution), see below
h	for negative half normal, $v = (2\pi)^{-1/2} - \|u\|$
e	for centered lognormal, $v = \text{Exp}(u) - \text{Sqr(e)}$
s	for Johnson S_b, $v = \text{Exp}(u) / [1 + \text{Exp}(u)]$
l	for lognormal, see below

The v_i above is a random draw from the indicated population. In cases *c, n, h, e, s,* and *l,* the draw v_i is the indicated transformation of a draw from the standard normal population. The negative half normal is a random variable with half normal density that is constrained to be less than zero. The Johnson S_b random variable ranges from zero to one. The centered lognormal variable ranges from -1.649 to +∞. It has the long tail of the lognormal distribution, but is shifted so as to have mean zero. In cases *u* and *t*, the draw is a transformation of a standard uniform, U(0,1) variable. For the tent distribution, the transformation is

$$v_i \ = \ 1(u < .5)[(2u)^{1/2} - 1] + 1(u > .5)[1 - (2(1 - u))^{1/2}]$$

This variable's density is a symmetric tent shape with mode at zero and which has support -1 to +1. For example, as shown earlier, the familiar random effects model is specified with **; Fcn = one(n)**. Note, there is no default distribution. You must specify one of the preceding (or a modification of it as shown below) in parentheses with the name. This is how you indicate that a parameter is to be treated as random in the model. Note, again, that the specification of the model (thus far) is

$$\beta_{k,i} \ = \ \beta_k^0 \ + \ \gamma_k v_{k,i}$$

so that your random variable has distribution with the shape of the selected variable, but is not constrained to the range of that distribution. For example, using **; Fcn = x(u)** produces a random parameter that has a uniform (flat) distribution with center at β_k and range $\beta_k \pm \sigma_k$. Likewise, the parameter with S_b distribution is not, itself, constrained to be in the 0,1 range.

Fixed Parameters

You can specify that a 'random' parameter has zero variance (is fixed) by using

; Fcn = name(c) or **name(*)**

This form specifies that the variance is zero, that is, only the mean varies. Of course, this is the same as not specifying the parameter as random, with one exception. If you have specified a hierarchical structure for the mean, then the covariates will still enter the mean of the variable. This is a way to build up a hierarchical structure for any model. Thus,

; Rpm = z1, z2
; Fcn = x(c)

specifies that the index function is of the form $\beta_i x = (\beta_1^0 + \delta_{1,1}z_1 + \delta_{1,2}z_2)x$. (This builds interaction terms.) Another form of this may be used to fix the mean of the parameter to a fixed value;

variable name(*,value) = a fixed parameter with zero variance and mean equal to value

(This is the same as **(dist|value)** in *NLOGIT* Version 6.) Thus, the parameter is nonrandom, and is fixed at the specified value. This differs from the previous specification in that with **(c)**, the mean is a free parameter and the variance equals zero, while with **(*,value)**, the mean is constrained to the given value with the variance also fixed at zero. This device provides a way to fix a parameter in the model. The **; Rst = list** specification would normally be used to do that, but **; Rst = list** is extremely difficult to use in this setting because there are so many parameters and during model setup, the parameter vector is reordered in a way that may not be easily predictable in a complicated specification. Note that this construction will be problematic if you have specified **; Rpm = variables,** as the specification is

$$\beta_{k,i} = \beta_k^0 + \boldsymbol{\delta}_k'\mathbf{z}_i + \gamma_k v_{k,i}.$$

and this form only allows you to fix β_k^0

Restricting the Range of a Parameter

You may specify that the parameter is lognormally distributed. This variant on the model will force a coefficient to be positive, and will also impose a particular form on the distribution of parameters across individuals. Use **; Fcn = name(l)** to request a lognormally distributed coefficient. In this case, for the particular random parameter β_{ik}, we will have

$$\beta_{ik} = \exp(\boldsymbol{\beta}_k^0 + \boldsymbol{\delta}_k'\mathbf{z}_i + \gamma_k v_{ik}).$$

Do note, if the coefficient is negative in an unrestricted model, forcing it to be positive may not work very well, or at all. This does, however, change the distribution of random parameters across individuals. A caution about this specification; it is often slow to converge, and frequently is inestimable. The assumption of lognormality is a strong one. Also, if the parameter you specified to be lognormally distributed tends in the sample to be negative, this will be an invalid restriction that will probably be revealed by nonconvergence. You can anticipate this if you fit the model as a fixed parameter model, and this parameter shows up as 'significantly' negative.

The triangular distribution can also be useful for restricting parameters. A special case of the triangular distribution specification is

variable name(t ,*) = a parameter that ranges from 0 to 2 β.

The triangular distribution is now restricted to one side of zero. The range is 0 to $2\beta^0$ in either the positive or negative direction, depending on the estimate of β^0. Figure R24.1 shows the implied model for the underlying parameter with this specification when the estimated parameter is 1.375. (The reported estimated 'scale factor' will also equal this value.)

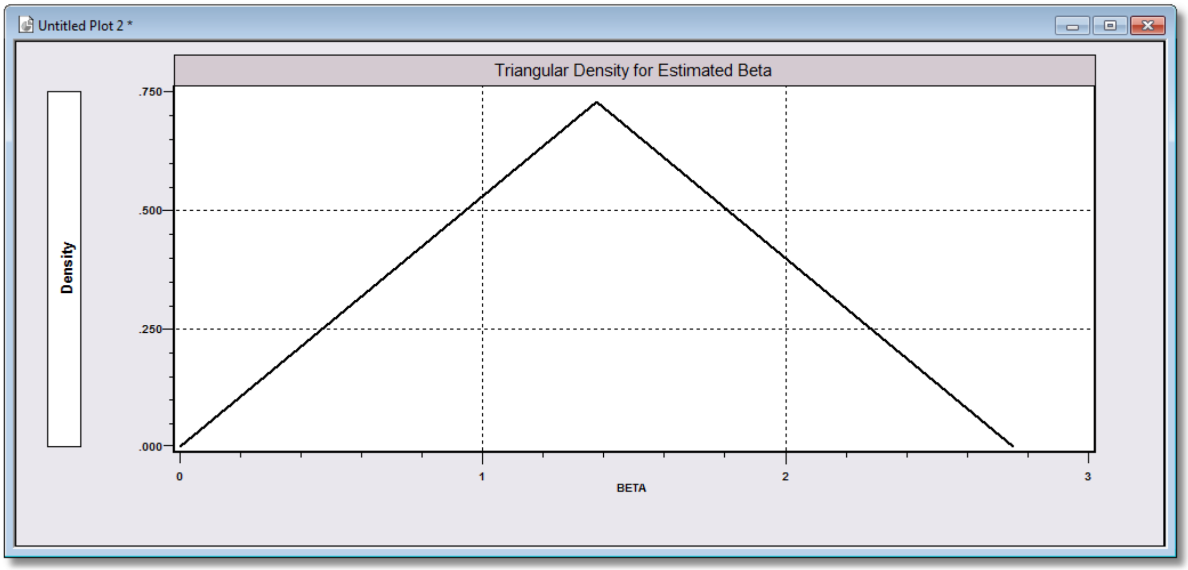


Figure R24.1 Estimated Constrained Triangular Distribution

Note that this specification will also be problematic if you have specified **; RPM = variables**.

Fixing the Mean

Other forms for random parameters are

variable name(type,value) = parameter with mean fixed at the value but a free variance

This form specifies that the mean of the parameter is fixed at *value* but the standard deviation is free. This forces the distribution of the random parameter to be centered at fixed mean *value*. For example, an interesting form is

; name(n, 0)

which defines a normally distributed parameter with mean zero. This is a type of random effects model.

The final two specifications modify the variables that enter the mean, that is the $\mathbf{\Delta z}_i$ term in $\boldsymbol{\beta}_i = \boldsymbol{\beta}^0 + \mathbf{\Delta z}_i + \mathbf{\Gamma v}_i$. You would have specified **; RPM = z** list. Then,

variable name(dist | #) = fixed mean parameter

specifies that the z_i variables do not appear in the mean of this specific parameter. This affects only the specific parameter of this specification. Formally, this specification constrains the indicated row of $\mathbf{\Delta}$ to be a row of zeros. You may also specify particular variables to appear in the mean with

variable name(dist | # pattern)

The pattern is ones and zeros to put zeros in cells in the row of $\mathbf{\Delta}$. (This is the same as in *NLOGIT*.) For example, in

; RPM = z1,z2
; Fcn = x1(n | # 01), x2 (n | # 10)

the specification makes the mean of the coefficient on $x1$ a function of $z2$, but not $z1$ and the mean of the coefficient on $x2$ a function of $z1$ but not $z2$.

Correlated Parameters

The default specification is for $\mathbf{\Gamma}$ to be a diagonal matrix, which implies that the random parameters are uncorrelated. You can specify correlated parameters with

; Cor

Your estimated model will now contain estimates of the triangular matrix as well as the implied covariance matrix of the parameters, deduced as $\mathbf{\Gamma\Gamma'}$. Note, although this is permissible with any specification of any model, the meaning of the estimated model is ambiguous if you have mixed distributions in your specifications. The correlation is induced by creating $\mathbf{v}_i$, then using $\mathbf{\Gamma v}_i$ to create the parameters. Without the correlation, your specified parameter has the indicated distribution. But, for example, if you specify one parameter (the first) normal and the second uniform, your second parameter will actually be composed as

$$\beta_{2i} = \beta_2 + \gamma_{12}v_{i1} + \gamma_{22}v_{i2},$$

that is, the sum of a normal and a uniform. Again, this is not precluded by the program, and you can fit such a model, but its meaning is a bit ambiguous.

Time Variation in Parameters

As noted, you can build some autocorrelation into your model by specifying that model for $\mathbf{v}_i$. To request this, use

; AR1

This specification of the model adds an autocorrelation parameter, ρ_i, for each random parameter. The generating mechanism for random parameters with this switch turned on is

$$\beta_{ki,t} = \beta_k^0 + \boldsymbol{\delta}_k'\mathbf{z}_i + \gamma_k v_{ki,t},$$
$$v_{ik,t} = \rho_k v_{ik,t\text{-}1} + u_{ik,t}$$

and $u_{ik,t}$ takes the distribution specified in the **name(dist)** specification. The distribution may be any of those listed earlier.

The earlier caution about mixing distributions is repeated, more emphatically here. In the autocorrelation model, the v_{it}s are generated by the autocorrelation scheme, then mixed by $\mathbf{\Gamma}$, so the same ambiguity arises. We advise that when using this form, you specify uncorrelated parameters.

Draws for the Simulated Log Likelihood – Number and Type

This model is fit by maximum simulated likelihood, as described below. The number of draws for the simulator is an important element of the estimator. If you specify too few, it is difficult to argue for consistency of the estimator. A few hundred is the norm. If you specify a very large number, estimation will take a long time. Estimation time is roughly linear in *R*. The default value is $R = 100$. You can change this with

; Pts = R (number of replications)

There is a body of theory relevant to this parameter (see Greene 2012, Chapter 15), but it states only that $R/N^{1/2}$ should diverge – *R* should increase faster than root-*N* – for the MSL estimator to be consistent. This does not state what *R* should be, however. Also, we note the important alternative to random draws, Halton sequences, makes much of this moot. To request Halton sequences instead of pseudorandom draws, use

; Halton

Halton sequences are discussed in detail below. We do suggest, when doing specification searches or experimental work, you can set *R* to a small value such as 25, and the estimator will perform satisfactorily. For 'production' work to generate final results, we do suggest several hundred draws.

Common Random Term in Random Parameter Models

In the RPM model, each random parameter, β_{ik}, has associated with it a random term, v_{ik}, that is specific to that parameter (and individual). It might be desired to have all parameters be functions of the same v_i. (That is the central feature of the Alvarez, Arias and Greene (2006) frontier model discussed in Section E64.11, for example.) To request this, you need add only

; Common

If you have specified multiple random parameters – this option has no effect unless you have – then the first specification in the list controls what the common random term will be.

Other Standard Specifications and Options

Finally, the following options operate the same as in the fixed parameters cases. Use

; Keep = name	to retain fitted values
; Res = name	to retain residuals
; Prob = name	to retain fitted probabilities for observed outcome
; Partial Effects	same as **; Marginal Effects**
; List	to display of predicted values (only if T_i is $\leq$ 10 for all *i*)
; Maxit = **n**	to set maximum iterations
; Wts = name	to specify weights – assumed the same for all periods if panel data

and other controls of optimization, such as setting the convergence criteria. Lagrange multiplier statistics will be difficult to obtain in some cases because the starting values for the iterations vary greatly from model to model, and will often not conform to what one might expect the 'restricted' model to be. In general, **; Maxit = 0** will produce ambiguous results at best.

R24.4 The Parameter Vector and Starting Values

Starting values for the iterations are generally obtained by fitting a basic model without random parameters – least squares for the linear model, tobit, etc., maximum likelihood probit, logit, Poisson, or exponential, and so on. In some cases, such as the stochastic frontier model, you must provide the starting values by fitting the restricted model. Every model description in the chapters to follow will specify how to estimate the RP model for that framework. Other RP parameters are set to zero. Thus, the initial results in the output for these models will often be the basic models discussed in the succeeding sections.

You may provide your own starting values for the parameters with

; Start = ... the list of values for all parameters in the model

The parameter vector is laid out as follows, in this order (where we introduce the new symbol, ϕ, to avoid some confusion)

$\phi_1, ..., \phi_{K1}$	are the K_1 nonrandom parameters (this is $\boldsymbol{\beta}_1$),
$\beta_1^0,...,\beta_{K2}^0$	are the K_2 means of the distributions of the random parameters,
$\sigma_1,\sigma_2,...,\sigma_{K2}$	are the K_2 scale parameters for the distributions of the random parameters.

These are the essential parameters. There are three optional parts, some or all of which may also be in the parameter vector. If you have specified that parameters are to be correlated, then the σs are followed by the below diagonal elements of $\boldsymbol{\Gamma}$. (The σs are the diagonal elements.) If there are two random parameters, then there is one below diagonal element. If there are three random parameters, then there are three below diagonal elements, and so on. The number of elements in total in this part of the parameter vector will be $K_2(K_2 - 1)/2$. These are supplied rowwise, so that this part of the parameter list will be

$$\boldsymbol{\Gamma} \quad = \quad \gamma_{21}, \gamma_{31}, \gamma_{32}, ..., \gamma_{K2,K2-1}$$

If you have specified heterogeneity variables with **; Rpm = z list**, then the elements of $\boldsymbol{\Gamma}$ are followed by the K_2 rows of $\boldsymbol{\Delta}$, each of which has M elements,

$$\boldsymbol{\Delta} \quad = \quad \delta_{11}, \delta_{12}, ..., \delta_{1M}, ..., \delta_{K2,1},...,\delta_{K2,M}.$$

Finally, if you have specified the model with autocorrelation, the K_2 autocorrelation coefficients will follow:

$$\boldsymbol{\rho} \quad = \quad \rho_1, \rho_2, ..., \rho_{K2}.$$

Consider an example: The model specifies:

; RPM = z1,z2
; Rhs = one,x1,x2,x3,x4 ? base parameters $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5$
; Fcn = one(n),x2(n),x4(n)
; Cor

Then, after rearranging to put the nonrandom parameters first, the model becomes

Variable	**Parameter**
$x1$	α_1
$x3$	α_2
one	$\beta_1^0 + \sigma_1 v_{i1} + \delta_{11} z_{i1} + \delta_{12} z_{i2}$
$x2$	$\beta_2^0 + \sigma_2 v_{i2} + \gamma_{21} v_{i1} + \delta_{11} z_{i1} + \delta_{12} z_{i2}$
$x4$	$\beta_3^0 + \sigma_3 v_{i3} + \gamma_{31} v_{i1} + \gamma_{32} v_{i2} + \delta_{11} z_{i1} + \delta_{12} z_{i2}$

and the parameter vector would be

$$\boldsymbol{\theta} = \phi_1, \phi_2, \beta_1^0, \beta_2^0, \beta_3^0, \sigma_1, \sigma_2, \sigma_3, \gamma_{21}, \gamma_{31}, \gamma_{32}, \delta_{11}, \delta_{12}, \delta_{21}, \delta_{22}, \delta_{31}, \delta_{32}.$$

You may use **; Rst** and **; CML** to impose restrictions on the parameters. Use the preceding as a guide to the arrangement of the parameter vector.

The variances of the underlying random variables are given earlier, 1 for the normal distribution, 1/3 for the uniform, and 1/6 for the tent distribution. The σ parameters are only the standard deviations for the normal distribution. For the other two distributions, σ_k is a scale parameter. The standard deviation is obtained as $\sigma_k/\sqrt{3}$ for the uniform distribution and $\sigma_k/\sqrt{6}$ for the triangular distribution. When the parameters are correlated, the implied covariance matrix is adjusted accordingly. The correlation matrix is unchanged by this.

R24.5 Individual Specific 'Estimates'

The random parameter models save the matrices, variables and scalars that are usual for the particular models – there is some difference across the model types. An additional result that is produced by the RP model for all the models is the conditional estimates of $E[\boldsymbol{\beta}_i|\, data_i, \hat{\boldsymbol{\theta}}]$ that can be produced for each individual (group) in the sample. Request this matrix with

; Parameters

This save matrices

beta_i	with individual specific estimates of $E[\boldsymbol{\beta}_i|\, data_i, \hat{\boldsymbol{\theta}}]$
sdbeta_i	= estimates of standard deviations of $[\boldsymbol{\beta}_i|\, data_i, \hat{\boldsymbol{\theta}}]$.

Each matrix has one column for each random parameter in the model. They do not contain the fixed parameter estimates. The next two sections provide details and examples of this computation.

The two matrices have

Rows	= number of individuals in the sample,
Columns	= number of random parameters in the model.

Each matrix is limited to 50,000 cells, so if your model with your sample size exceeds that, the matrices are filled by observations until they run out of room. You can restrict the matrices to retain only certain columns from the predicted values as follows: When you set up the function types with **; Fcn = list of types**, place a dot before the type specifications you wish to save. For the example above, which has

; Fcn = one(n),x2(n),x4(n),

This will create three column matrices, with a column for each coefficient. If it is changed to

```
; Fcn = one(.n),x2(n),x4(.n)
```

then, the matrices will have only two columns, containing the estimates of the first and third parameters.

The matrices can be moved to the data area by setting up a template. Suppose the matrix has *N* rows and four columns. If the sample is 1000 individuals, the matrix *beta_i* will have 1000 rows and four columns. You can use

```
CREATE      ; b1 = 0 ; b2 = 0 ; b3 = 0 ; b4 = 0 $
NAMELIST    ; rpbeta = b1,b2,b3,b4 $
CREATE      ; rpbeta = beta_i $
```

This creates four variables in the first 1,000 rows in the current sample. This will probably be the first 1,000 rows in the data area, but might not be.

Before documenting the computation, three notes are important regarding these statistics. (The literature on random parameter models, especially the Bayesian part of it, is particularly loose on these points. An exception is Train (2009), to which we refer the interested reader.) First, the computation is performed at the estimated values of the population parameters, $\hat{\boldsymbol{\theta}}$. As such, the statistics do not take into consideration the sampling error in the estimate of θ. Second, the computation described here calculates the mean and standard deviation of the *conditional* distribution of $\boldsymbol{\beta}_i$. That is, we describe how to estimate $E[\boldsymbol{\beta}_i|\ data_i,\ \hat{\boldsymbol{\theta}}]$ and $Std[\boldsymbol{\beta}_i|\ data_i,\ \hat{\boldsymbol{\theta}}]$, where '$data_i$' denotes all the information in hand about individual *i* including the dependent variable and $\hat{\boldsymbol{\theta}}$ is the estimate of the population parameters. The conditional distribution for person *i* is the distribution of $\boldsymbol{\beta}$ within the subpopulation of people who, if they faced the same choice situations as *i* with the same variables, would make the same choices as *i*. Since individuals with different coefficients can have the same outcomes when they face the same situations, there is a distribution of coefficients within this subpopulation of individuals. The conditional distribution is this distribution. Third, the conditional mean is a consistent estimator of $\boldsymbol{\beta}_i$ *only* if the number of observations in the group, T_i, rises without bound along with the sample size, *N*. In most situations, the number of outcomes for each sampled individual is naturally limited and cannot rise without bound. (For example, if T_i is the number of times a person buys a new car, it is logically impossible for T_i to rise without bound.) In these cases, the conditional mean of $\boldsymbol{\beta}_i$ is not a consistent estimator of individual *i*'s true coefficients. To focus the idea, consider that two individuals *q* and *r* in the sample could have identical right hand side variables $\mathbf{x}_i$, identical covariates, $\mathbf{z}_i$ and identical outcomes, y_i. (Certainly this could occur in a simple model with only a few variables in it.) Nonetheless, being two different individuals, they could (indeed, given the distribution is continuous, they would) be characterized by two different parameter vectors, $\boldsymbol{\beta}_q$ and $\boldsymbol{\beta}_r$. Given that all the measured information is the same for the two, both parameter vectors are draws from the same conditional distribution, $p(\boldsymbol{\beta}_i|\ \mathbf{x}_i,\ \mathbf{z}_i, y_i,\ \hat{\boldsymbol{\theta}})$. We will estimate the mean of this distribution, $E(\boldsymbol{\beta}_i|\ \mathbf{x}_i,\ \mathbf{z}_i, y_i,\ \hat{\boldsymbol{\theta}})$, which will then characterize both individuals the same. This mean is not a consistent estimator of either $\boldsymbol{\beta}_q$ or $\boldsymbol{\beta}_r$. To the point, increasing sample sizes will produce a better estimate of $\boldsymbol{\theta}$, but as long as the number of choice situations faced by each person is fixed, $E(\boldsymbol{\beta}_i|\ \mathbf{x}_i,\ \mathbf{z}_i, y_i,\ \hat{\boldsymbol{\theta}})$ will be the same for $i = q$ and $i = r$ even though $\boldsymbol{\beta}_q \neq \boldsymbol{\beta}_r$. On the other hand, if somehow the number of choice situations faced by each person rose without bound, then eventually the two people would make different choices when facing the same right hand side variables, such that y_i would differ for $i = q$ and $i = r$. With different values of y_i the conditional means $E(\boldsymbol{\beta}_i|\ \mathbf{x}_i,\ \mathbf{z}_i, y_i,\ \hat{\boldsymbol{\theta}})$ for the two people would differ; and with enough choice situations faced by each person, the conditional means for the two people would converge to each person's own true coefficients. However, as this explanation hopefully makes clear, this convergence to the true coefficients for each person only occurs when the number of choice situations faced by each person rises without bound. In the vast majority of applications, the number of choice situations faced by each person cannot rise without bound and, in fact, is quite small, such that the conditional mean is not a consistent estimator of the person's true coefficients. This line of logic applies both to the conditional means estimated here and to the posterior means estimated in Bayesian analysis.

Estimates of Conditional Means

The estimates of the model parameters provide the unconditional estimates of the parameter vectors. The precise construction differs from one model to the next, but in general, that estimate is the prior mean,

$$\hat{\boldsymbol{\alpha}}_i = \left[\hat{\boldsymbol{\beta}}_1, \hat{\boldsymbol{\beta}}_2^0 + \hat{\boldsymbol{\Delta}}\mathbf{z}_i, \hat{\boldsymbol{\theta}}\right]$$

This estimator uses the aggregate of the information in the sample and, if present, information contained in $\mathbf{z}_i$. However, we can also form a person specific conditional estimator of the second component. (The first and third are viewed as constants, so prior and posterior are the same.) Discussion of this computation may be found in Train (2009). The estimator of the conditional mean of the distribution of the random parameters – conditioned on the person specific data – is

$$\overset{\wedge}{\boldsymbol{\alpha}}_i = \hat{E}[\boldsymbol{\alpha}_i \mid data_i] = \frac{(1/R)\sum_{r=1}^{R} \hat{f}_{ir} \overset{\wedge}{\boldsymbol{\alpha}}_{ir}}{(1/R)\sum_{r=1}^{R} \hat{f}_{ir}}, \quad \hat{\boldsymbol{\alpha}}_{ir} = \left[\hat{\boldsymbol{\beta}}_1, \hat{\boldsymbol{\beta}}_2^0 + \hat{\boldsymbol{\Delta}}\mathbf{z}_i + \hat{\boldsymbol{\Gamma}}\mathbf{v}_{ir}, \hat{\boldsymbol{\theta}}\right]$$

where summation over R is the within observation summation over the simulation replications, and the weights are the joint densities that enter (in log form) the log likelihood,

$$\hat{f}_{ir} = \prod_{t=1}^{T_i} f(y_{it}, \mathbf{x}_{it}, \mathbf{z}_i, \mathbf{v}_{ir}, \text{all estimated structural parameters}).$$

Use the **; Parameters** specification in your model command to request this computation. This will save the matrix named *beta_i* containing the estimates of the second component, $E[\boldsymbol{\beta}_2|data_i]$. (Since the nonrandom components are constant, the averages will just equal those constants.) Note, this matrix may be quite large, as there is one vector for each individual in the sample – each person is a row in this matrix. We also estimate the standard deviation of this distribution by estimating for each random parameter,

$$\hat{E}[\beta_{i,k}^2 \mid data_i] = \frac{(1/R)\sum_{r=1}^{R} \hat{f}_{ir}\hat{\beta}_{ir,k}^2}{(1/R)\sum_{r=1}^{R} \hat{f}_{ir}}$$

then computing the square root of the estimated variance,

$$\sqrt{\hat{E}[\beta_{i,k}^2 \mid data_i] - \left(\hat{E}[\beta_{i,k} \mid data_i]\right)^2}\ .$$

These are the values that appear in the matrix *sdbeta_i*. The application below suggests how one might use this information.

Random Parameters Fitted Values

The fitted values routines have been changed to use the simulated random parameters rather than the global means. The parameter vector in the random parameters model is

$$\boldsymbol{\beta}_i = \boldsymbol{\beta}^0 + \boldsymbol{\Delta}\mathbf{z}_i + \boldsymbol{\Gamma}\mathbf{v}_i.$$

The estimators use the conditional (posterior) estimator, $E[\boldsymbol{\beta}_i|data_i, \hat{\boldsymbol{\theta}}]$.

R24.6 Application

We will illustrate a few useful features of the RP model with the data set used in several earlier applications. The German health care data examined in Section E2.4 and several later examples provides a straightforward panel data set. For this purpose, we will examine a small subsample of the full data set. The commands for estimating a Poisson regression for number of doctor visits are

```
SAMPLE      ; 1-5000 $
REJECT      ; _groupti < 7 $
POISSON     ; Lhs = docvis ; Rhs = one,female,age,hhninc,hhkids,educ
            ; RPM
            ; Fcn = one(n),age(n),hhninc(n)
            ; Correlated parameters
            ; Pds = 7 ; Pts = 25
            ; Halton draws
            ; Parameters  $
```

We obtained the following estimation results. The initial results are the starting values obtained by fitting the model with all nonrandom parameters. Note that the Rhs variables have been reordered before the beginning of estimation.

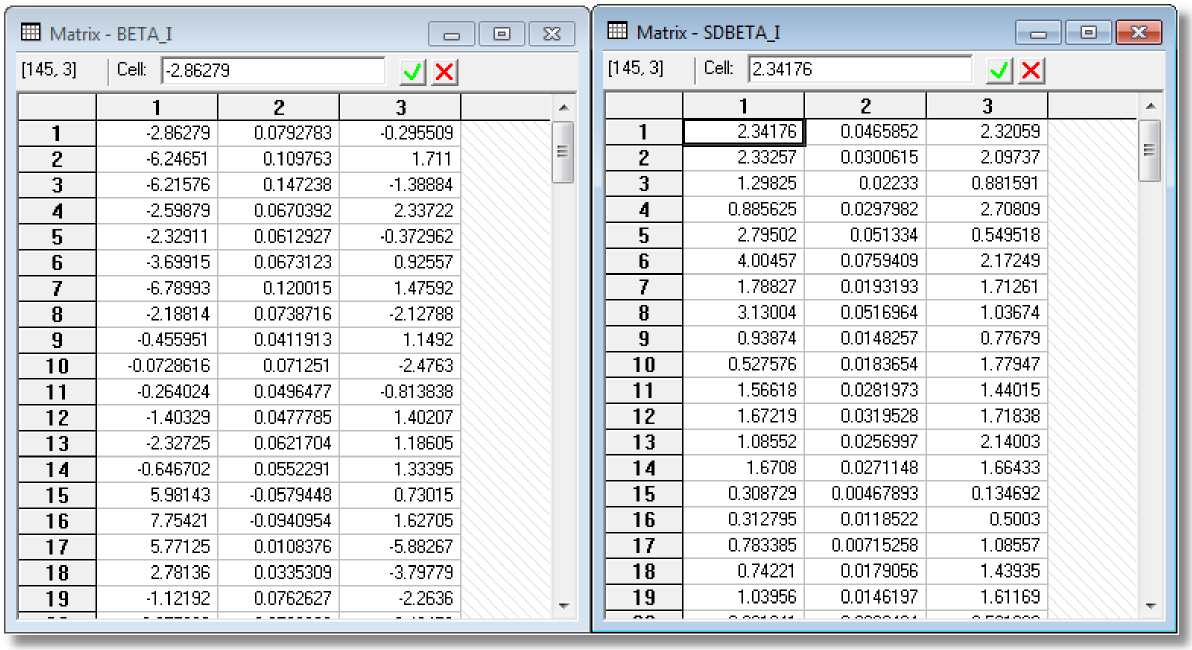
```
-----------------------------------------------------------------------------
Poisson  Regression Start Values for DOCVIS
Dependent variable                DOCVIS
Log likelihood function    -11538.30728
Estimation based on N =   1015, K =   6
Inf.Cr.AIC  =23088.615 AIC/N =   22.747
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
  DOCVIS|  Coefficient      Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
  FEMALE|    .32063***     .03269     9.81  .0000      .25656     .38470
  HHKIDS|   -.09901***     .03809    -2.60  .0093    -.17367    -.02436
    EDUC|   -.11206***     .00984   -11.39  .0000    -.13134    -.09277
Constant|   1.94456***     .15840    12.28  .0000    1.63410    2.25502
     AGE|    .01441***     .00215     6.70  .0000      .01020     .01863
  HHNINC|   -.18130*       .10435    -1.74  .0823    -.38582     .02323
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

The next set of results is the estimates of the structural parameters of the random parameters model. The parameter estimates given include β_1, β_2^0, the diagonal elements of Γ, then the below diagonal elements of Γ. The matrix given as *var_beta* is computed as $\mathbf{\Gamma\Gamma'}$. The matrix *s.d_beta* is the vector of square roots of the diagonal elements of $\mathbf{\Gamma\Gamma'}$. Finally, the correlation matrix is derived from the covariance matrix.

```
Normal exit:  35 iterations. Status=0, F=    2952.749

-----------------------------------------------------------------------------
Random Coefficients  Poisson  Model
Dependent variable               DOCVIS
Log likelihood function     -2952.74858
Restricted log likelihood  -11538.30728
Chi squared [   6 d.f.]     17171.11739
Significance level               .00000
McFadden Pseudo R-squared      .7440917
Estimation based on N =   1015, K =  12
Inf.Cr.AIC  = 5929.497 AIC/N =    5.842
Sample is  7 pds and    145 individuals
POISSON regression model
Simulation based on  25 Halton draws
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCVIS|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Nonrandom parameters
  FEMALE|     .25071***      .02936     8.54  .0000      .19316    .30826
  HHKIDS|    -.21606***      .02781    -7.77  .0000    -.27057   -.16155
    EDUC|    -.08131***      .00891    -9.12  .0000    -.09878   -.06384
        |Means for random parameters
Constant|     .94045***      .13317     7.06  .0000      .67943   1.20146
     AGE|     .03588***      .00184    19.55  .0000      .03229    .03948
  HHNINC|    -.93486***      .09543    -9.80  .0000   -1.12189   -.74783
        |Diagonal elements of Cholesky matrix
Constant|    4.63750***      .12811    36.20  .0000    4.38642   4.88858
     AGE|     .02874***      .00079    36.42  .0000      .02720    .03029
  HHNINC|    1.29611***      .03842    33.73  .0000    1.22081   1.37142
        |Below diagonal elements of Cholesky matrix
lAGE_ONE|    -.05637***      .00260   -21.67  .0000    -.06147   -.05127
lHHN_ONE|   -1.54333***      .13804   -11.18  .0000   -1.81389  -1.27277
lHHN_AGE|   -2.46124***      .09005   -27.33  .0000   -2.63773  -2.28475
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

Implied covariance matrix of random parameters

Var_Beta|             1             2             3
--------+------------------------------------------
       1|       21.5064      -.261432      -7.15719
       2|      -.261432     .00400406      .0162620
       3|      -7.15719      .0162620       10.1195
```

```
Implied standard deviations of random parameters

S.D_Beta|             1
--------+--------------
       1|       4.63750
       2|      .0632777
       3|       3.18111

Implied correlation matrix of random parameters

Cor_Beta|             1             2             3
--------+------------------------------------------
       1|       1.00000      -.890890      -.485154
       2|      -.890890       1.00000      .0807875
       3|      -.485154      .0807875       1.00000
```

The **; Correlated Parameters** specification requests computation of the two matrices of conditional means. These appear as follows:



Matrix - BETA_I

[145, 3] Cell: -2.86279

	1	2	3
1	-2.86279	0.0792783	-0.295509
2	-6.24651	0.109763	1.711
3	-6.21576	0.147238	-1.38884
4	-2.59879	0.0670392	2.33722
5	-2.32911	0.0612927	-0.372962
6	-3.69915	0.0673123	0.92557
7	-6.78993	0.120015	1.47592
8	-2.18814	0.0738716	-2.12788
9	-0.455951	0.0411913	1.1492
10	-0.0728616	0.071251	-2.4763
11	-0.264024	0.0496477	-0.813838
12	-1.40329	0.0477785	1.40207
13	-2.32725	0.0621704	1.18605
14	-0.646702	0.0552291	1.33395
15	5.98143	-0.0579448	0.73015
16	7.75421	-0.0940954	1.62705
17	5.77125	0.0108376	-5.88267
18	2.78136	0.0335309	-3.79779
19	-1.12192	0.0762627	-2.2636

Matrix - SDBETA_I

[145, 3] Cell: 2.34176

	1	2	3
1	2.34176	0.0465852	2.32059
2	2.33257	0.0300615	2.09737
3	1.29825	0.02233	0.881591
4	0.885625	0.0297982	2.70809
5	2.79502	0.051334	0.549518
6	4.00457	0.0759409	2.17249
7	1.78827	0.0193193	1.71261
8	3.13004	0.0516964	1.03674
9	0.93874	0.0148257	0.77679
10	0.527576	0.0183654	1.77947
11	1.56618	0.0281973	1.44015
12	1.67219	0.0319528	1.71838
13	1.08552	0.0256997	2.14003
14	1.6708	0.0271148	1.66433
15	0.308729	0.00467893	0.134692
16	0.312795	0.0118522	0.5003
17	0.783385	0.00715258	1.08557
18	0.74221	0.0179056	1.43935
19	1.03956	0.0146197	1.61169

Figure R24.2 Estimated Conditional Means

Note, the sample contains 145 individuals, so that is the number of rows in the matrices. There is a considerable amount of variation across individuals in both means and standard deviations. To explore this, we will construct a plot that shows this variation.

The following extracts the parts of the two matrices and constructs an interesting figure.

```
SAMPLE      ; 1-145 $
MATRIX      ; b_inc = beta_i (1:145, 2:2) $
MATRIX      ; sb_inc = sdbeta_i (1:145, 2:2) $
CREATE      ; beta_inc  = b_inc $
CREATE      ; sbeta = sb_inc $
CREATE      ; lower = beta_inc - 2*sbeta
            ; upper = beta_inc + 2*sbeta $
CREATE      ; person = Trn(1,1) $
PLOT        ; Lhs = person
            ; Rhs = lower,upper ; Centipede
            ; Title = 95% Probability Intervals for Beta(Income)
            ; Yaxis =  Range
            ; Endpoints = 0,150 ; Bars = 0 $
```

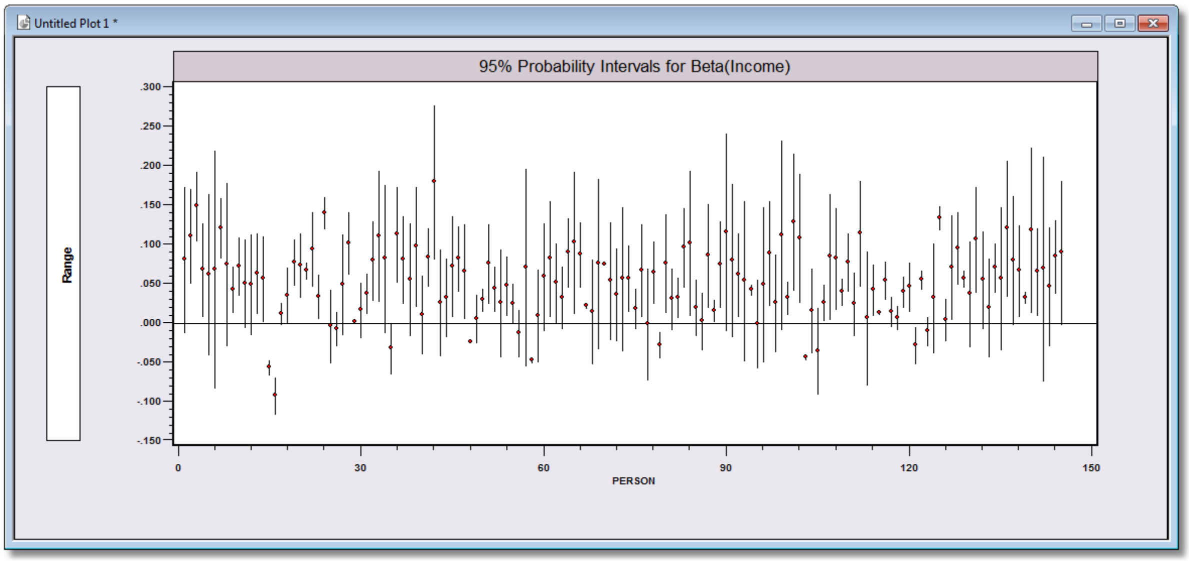


Figure R24.3 Confidence Intervals for Conditional Means

As noted earlier, the conditional means (the dots in the centipede plot) are not actually estimates of β_i. However, β_i is a draw from the conditional distribution $P(\beta_i|data_i,\theta)$. The spikes in the figure above represent estimates of a range of this density that should capture a large proportion of the mass of the distribution. In general, a mean plus two standard deviations will capture at least 95% of any but the most pathological distribution. Thus, subject to a couple caveats we'll note, the lines in the figure above do represent confidence regions for β_i. The caveats are: first, the estimates are based on the estimates of the structural parameters, $\hat{\theta}$, not the actual parameters, θ. Thus, by not accounting for this variation, the intervals above are too narrow. As can be seen in the estimation results, the structural parameters are estimated quite precisely. Second, the precise shape of the distribution is not known. If it is asymmetric, then the preceding might be too narrow. This would be a minor consideration, however. Overall, then, we might reasonably consider the figure above to display a confidence interval for the parameter β_i.

R24.7 Technical Details on Estimation of RP Models by Simulation

For purposes of this presentation, we will change slightly the mathematical form of the model. The log likelihood for the random parameters is formulated and maximized with the steps described below. This derivation will be lengthy. We have collected the results that are used in several modeling frameworks here in one place so that they may be easily accessed in one location in the manual. The identical mathematical results apply to all the models listed earlier.

The Theoretical Likelihood Function and Derivatives

The structure of the random parameters model is based on the conditional density (slightly abbreviated for the moment)

$$P(y_{it}|\,\mathbf{x}_{it},\,\boldsymbol{\beta}_i) \;=\; g(y_{it},\,\boldsymbol{\beta}_i{}'\mathbf{x}_{it}),\; i = 1,...,N,\; t = 1,...,T_i.$$

where $g(.)$ is the density discussed earlier. The model assumes that parameters are randomly distributed with possibly heterogeneous (across individuals) mean

$$E[\boldsymbol{\beta}_i|\,\mathbf{z}_i] \;=\; \boldsymbol{\beta} \;+\; \boldsymbol{\Delta}\mathbf{z}_i,$$

(the second term is optional – the mean may be constant),

$$\text{Var}[\boldsymbol{\beta}_i|\,\mathbf{z}_i] \;=\; \boldsymbol{\Sigma}.$$

By construction, then,

$$\boldsymbol{\beta}_i \;=\; \boldsymbol{\beta} \;+\; \boldsymbol{\Delta}\mathbf{z}_i \;+\; \boldsymbol{\Gamma}\mathbf{v}_i.$$

(Note that $\boldsymbol{\beta}_i$ could vary across time as well as individuals. This follows from the extension of the model to allow autocorrelation in the random terms. This is developed below.) The third term is simply the deviation of $\boldsymbol{\beta}_i$ from its theoretical mean. It is convenient to analyze the model in this fully general form at this point. One can easily accommodate nonrandom parameters just by placing rows of zeros in the appropriate places in $\boldsymbol{\Delta}$ and $\boldsymbol{\Gamma}$. The actual treatment is discussed in the preceding section.

NOTE: If there is no heterogeneity in the mean, and only the constant term is considered random – the model may specify that some parameters are nonrandom – then this model is equivalent to the random effects model discussed earlier.

The true log likelihood function is

$$\log L \;=\; \Sigma_i \; \log L_i$$

where $\log L_i$ is the contribution of the ith individual (group) to the total. Conditioned on $\mathbf{v}_i$, the joint probability for the ith group is

$$P[y_{i1},...,y_{iTi}\,|\,\mathbf{x}_{it,...,}\,\mathbf{z}_{i,}\mathbf{v}_i,\; t = 1,...,T_i] \;=\; \prod_{t=1}^{T_i} g(y_{it},\boldsymbol{\beta}_i'\mathbf{x}_{it})$$

Since $\mathbf{v}_i$ is unobserved, it is necessary to obtain the unconditional log likelihood by taking the expectation of this over the distribution of $\mathbf{v}_i$. Thus,

$$L_i | \mathbf{v}_i, t = 1,...,T_i = \prod_{t=1}^{T_i} g(y_{it}, \boldsymbol{\beta}_i'\mathbf{x}_{it}) .$$

Then, $$L_i = E_{\mathbf{v}i} [L_i | \mathbf{v}_i, t = 1,...,T_i] = \int_{\text{Range of } \mathbf{v}_i} g(\mathbf{v}_i, t = 1,...,T_i) \prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}_i'\mathbf{x}_{it}) \, d\mathbf{v}_i$$

(Note that this is a multivariate integral.) Then, finally,

$$\log L = \sum_{i=1}^{N} \log L_i$$

For convenience in what follows, let $\boldsymbol{\Theta}$ = the full vector of all parameters in the model. The likelihood function is maximized by solving the likelihood equations:

$$\frac{\partial \log L}{\partial \boldsymbol{\Theta}} = \sum_{i=1}^{N} \frac{\partial \log L_i}{\partial \boldsymbol{\Theta}} = \mathbf{0},$$

and note that these derivatives will likewise involve integration. For this estimator, the integration is done by Monte Carlo simulation. In general, we use the approximation strategy:

$$E_{\mathbf{v}i} [L_i | \mathbf{v}_i, t = 1,...,T_i] \approx \frac{1}{R} \sum_{r=1}^{R} L_i \,|\, (\mathbf{v}_{ir}, t = 1,...,T_i)$$

where $\mathbf{v}_{ir}$ is a set of T_i K_2-variate random draw from the joint distribution of $\mathbf{v}_i$. (I.e., it is a draw of a $T_i \times K_2$ random matrix. In the case of no autocorrelation, there is only one K_2-variate draw, which is then the same in all periods, in the fashion of a random effects model.) See Brownstone and Train (1999), Train (1998, 2009), and Revelt and Train (1998) for discussion. The approximation improves with increased R (this is under your control) and with increases in N, though the simulation variance which decreases with increases in R does not decrease with N.

Random Draws for the Simulations

The K_2 elements of $\mathbf{v}_{ir}$ are drawn as follows: We begin with a K_2 random vector $\mathbf{w}_{ir}$ that is

K_2 independent draws from the standard uniform [0,1] distribution

or

K_2 Halton draws from the *m*th Halton sequence, where m is the *m*th prime number in the sequence of K_2 prime numbers beginning with 2.

The Halton values are also distributed in the unit interval. They are described in detail below. This primitive draw is then transformed to the distribution specified in the **; Fcn** specification, as follows:

Uniform[-1,1]: $u_{k,ir} = 2w_{k,ir} - 1$

Tent [-1,1] $u_{k,ir} = \mathbf{1}(w_{k,ir} \leq .5)[\sqrt{2w_{k,ir}} - 1] + \mathbf{1}(w_{k,ir} > .5)[1 - \sqrt{2(1-w_{k,ir})}\ ]$

Normal[0,1] $u_{k,ir} = \Phi^{-1}(w_{k,ir})$

(Other transformations are listed at the beginning of this section.) This produces a K_2 vector, $\mathbf{u}_{ir}$. Finally, $\mathbf{v}_{ir}$ is obtained as follows:

1. No autocorrelation: $\mathbf{v}_{ir} = \mathbf{u}_{ir}$ for all t.

   In this case, $\mathbf{w}_{ir}$ is drawn once for the entire set of T_i periods, and reused. This is the standard 'random effect' arrangement, in which the effect is the same in every period. In this case,

   $$\mathbf{w}_{itr} = \mathbf{w}_{ir},\ \mathbf{u}_{itr} = \mathbf{u}_{ir},\ \text{and } \mathbf{v}_{itr} = \mathbf{v}_{ir},$$

2. AR1 model (autocorrelation): $v_{k,i1r} = [1/(1 - \rho_k^2)]\, u_{k,i1r}$

   $$v_{k,itr} = \rho_k v_{k,i,t-1,r} + u_{k,itr}$$

   This is the standard first order autocorrelation treatment, with the Prais-Winsten treatment for the first observation to avoid losing any observations due to differencing. For this case, $u_{k,ir}$ has been drawn for each period, $u_{k,itr}$, then used in the transformation immediately above to produce $\mathbf{v}_{itr}$.

In the preceding derivation, it is stated that $\mathbf{\Omega} = \mathbf{\Gamma\Gamma}'$ is the covariance matrix of $\mathbf{\Gamma v}_{itr}$. This is true for the standard normal case. For the other two cases, a further scaling is needed. The variance of the uniform [-1,1] is the squared width over 12, or 1/3, so its standard deviation is $1/\sqrt{3} = .57735$. The variance of the standardized tent distribution is 1/6. (Since this is a density with a discontinuous derivative, this takes a bit of derivation to show. It can be shown by partitioning the distribution. The density of u in this case is

$$f(u) = 2(1+u) \text{ for } u \leq 0 \text{ and } 2(1-u) \text{ for } u > 0.$$

The probability in each section is ½. The mean is obviously zero (by construction). The two conditional means are -1/3 and +1/3 for the left and right halves. The conditional variances can be found by simple integration to be 1/18 in each half. The variance equals the variance of the conditional mean plus the expected value of the conditional variance, which gives 1/9 for the former and 1/18 for the latter, which sum to 1/6. The standard deviation is therefore .40824. This implicit scaling is undone at the time the results are reported.

Controlling the Simulation

There are two parameters of the simulations that you can change. The number of points in the simulation is *R*. Authors differ in the appropriate value. Train (2009) recommends several hundred. Bhat (2001) suggests 1,000 as an appropriate value. The program default is 100. You can choose the value with

; Pts = number of draws, R

In order to replicate an estimation, you must use the same random draws. One implication of this is that if you give the identical model command twice in sequence, you will not get the identical set of results because the random draws in the sequences will be different. To obtain the same results, you must reset the seed of the random number generator with a command such as

```
CALC            ; Ran (seed value) $
```

We often use **Ran(12345)** before each of our examples, precisely for this reason. The specific value you use for the seed is not of consequence; any odd number will do.

In this connection, we note a consideration which is crucial in this sort of estimation. The random sequence used for the model estimation must be the same in order to obtain replicability. In addition, during estimation of a particular model, the same set of random draws must be used for each person every time. That is, the sequence $\mathbf{v}_{i1}$, $\mathbf{v}_{i2}$, ..., $\mathbf{v}_{iR}$ used for each individual must be the same every time it is used to calculate a probability, derivative, or likelihood function. (If this is not the case, the likelihood function will be discontinuous in the parameters, and successful estimation becomes unlikely.) One way to achieve this which has been suggested in the literature is to store the random numbers in advance, and simply draw from this reservoir of values as needed. Because *LIMDEP* is able to use very large samples, this is not a practical solution, especially if the number of draws is large as well. We achieve the same result by assigning to each individual, *i*, in the sample, their own random generator seed which is a unique function of the global random number seed, *S*, and their group number, *i*;

$$\text{Seed}(S,i) = S + 123.0 \times i, \text{ then minus 1.0 if the result is even.}$$

Since the global seed, *S*, is a positive odd number, this seed value is unique, at least within the several million observation range of *LIMDEP*.

Halton Draws and Random Draws for Simulations

The standard approach to simulation estimation is to use random draws from the specified distribution. As suggested above, good performance in this connection requires very large numbers of draws. The drawback to this approach is that with large samples and large models, this entails a huge amount of computation and can be very time consuming. Some authors have documented dramatic speed gains with no degradation in simulation performance through the use of a small number of Halton draws instead of a large number of random draws. Authors (e.g., Bhat (2001)) have found that a Halton sequence of draws with only one tenth the number of draws as a random sequence is often equally effective. To use this approach, add

; Halton

to your model command.

Conventional simulation based estimation uses a random number to produce a large number of draws from a specified distribution. The central component of the standard approach is draws from the standard continuous uniform distribution, $U[0,1]$. (*LIMDEP*'s random number generator is described in Appendix R4A.3.) Draws from other distributions are obtained from these draws by using transformations. In particular, where u_i is one draw from $U[0,1]$,

Normal [0,1]: $v_i = \Phi^{-1}(u_i)$

Uniform[-1,1]: $v_i = 2u_i - 1$

Tent: $v_i = \sqrt{2u_i} - 1$ if $u_i \leq 0.5$, $v_i = 1 - \sqrt{2u_i - 1}$ otherwise.

Given that the initial draws satisfy the assumptions necessary, the central issue for purposes of specifying the simulation is the number of draws. Results differ on the number needed in a given application, but the general finding is that when simulation is done in this fashion, the number is large. A consequence of this is that for large scale problems, the amount of computation time in simulation based estimation can be extremely long.

Procedures have recently been devised in the numerical analysis literature for taking 'intelligent' draws from the uniform distribution, rather than random ones. (See Train (1999, 2009) and Bhat (2001) for extensive discussion and further references.) These procedures appear vastly to reduce the number of draws needed for estimation (by a factor of 90% or more) and reduce the simulation error associated with a given number of draws. In one application of the method to be discussed here, Bhat (2001) found that 100 Halton draws produced lower simulation error than 1,000 random numbers.

The procedure described here is labeled Halton sequences. (See Train (1999).) The Halton sequence is generated as follows: Let r be a prime number larger than two. Expand the sequence of integers $g = 1,...$ in terms of the base r as

$$g = \sum_{i=0}^{I} b_i r^i \text{ where by construction, } 0 \leq b_i \leq r - 1 \text{ and } r^I \leq g < r^{I+1}.$$

The Halton sequence of values that corresponds to this series is

$$H(g) = \sum_{i=0}^{I} b_i r^{-i-1}$$

For example, using base 5, the integer 37 has $b_0 = 2$, $b_1 = 2$, and $b_3 = 1$. Then

$$H(37) = 2\times5^{-1} + 2\times5^{-2} + 1\times5^{-3} = 0.448.$$

The sequence of Halton values is efficiently spread over the unit interval. The sequence is not random as the sequence of pseudorandom numbers is.

The figures below show two sequences of Halton draws and two sequences of pseudorandom draws. The Halton draws are based on $r = 7$ and $r = 9$. The clumping evident in the first figure is the feature (among other others) that mandates large samples for simulations. We use the prime numbers in order beginning with 3. If a model requires K random draws, we use the first K prime numbers to generate the sequences. Within each series, the first 10 draws are discarded, as these draws tend to be highly correlated across different periods.

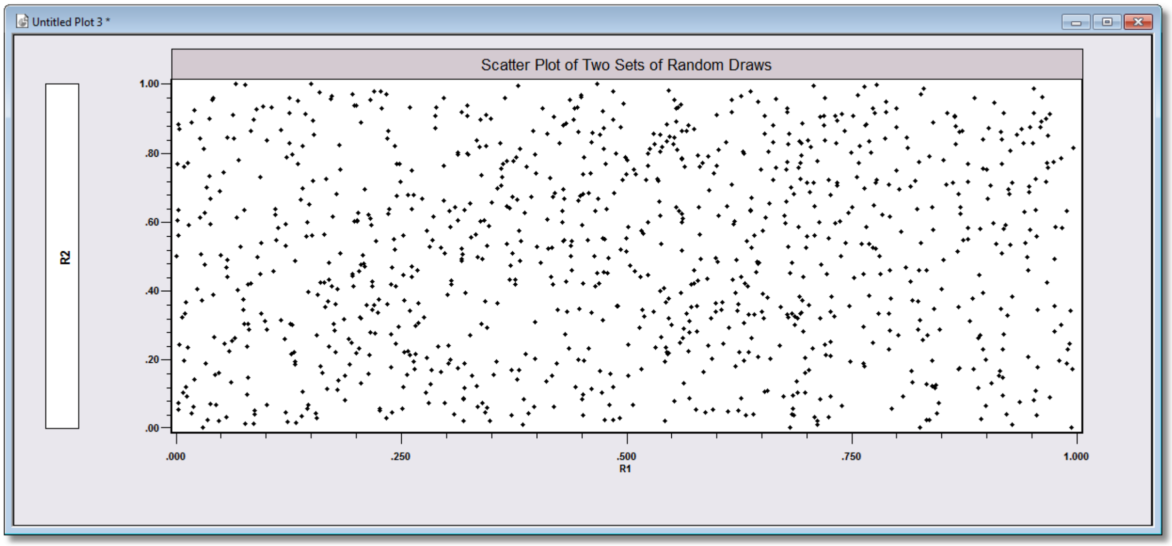


Figure R24.4 Bivariate Distribution of Random Uniform Draws

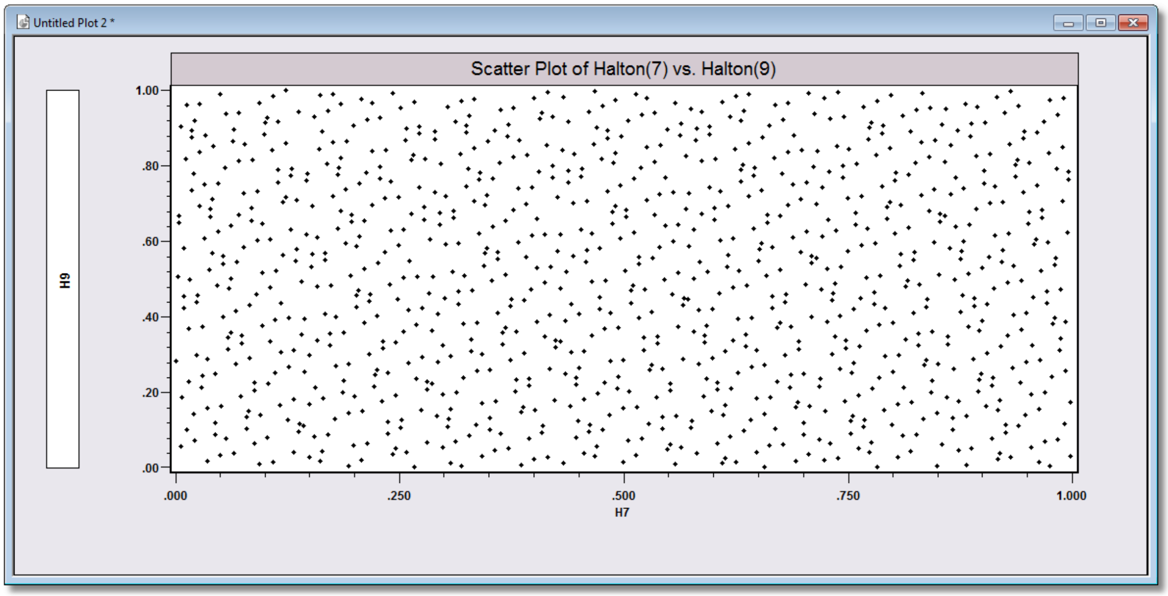


Figure R24.5 Bivariate Distribution of Halton (7) and Halton (9)

You can draw Halton sequences for your own purposes. The command

CREATE ; name = Hlt(prime number) $

generates a Halton sequence for the *n* integers, where *n* is your sample size, beginning at one. The descriptive statistics below show the behavior of the Halton sequences. Note, they are not random – they are not intended to be. The ACFs and PACFs reflect the underlying construction of the data. The descriptive statistics do show how the series covers the uniform distribution, which has mean .5 and standard deviation $1/\sqrt{12}$ = .288675.

```
Time series identification for H7
PACF is computed using Yule-Walker equations.
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Lag |  Autocorrelation Function     |Box/Prc|    Partial Autocorrelations   X
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  1 | .264*|           |***         | 69.88*| .264*|            |***        X
  2 |-.227*|        ***|            |121.60*|-.320*|       **** |           X
  3 |-.475*|      *****|            |347.03*|-.374*|       **** |           X
  4 |-.478*|      *****|            |575.07*|-.438*|      ***** |           X
  5 |-.235*|        ***|            |630.47*|-.485*|      ***** |           X
  6 | .252*|           |***         |693.93*|-.302*|        *** |           X
  7 | .978*|           |*********** |*******| .958*|            |***********X
  8 | .248*|           |***         |*******|-.690*|   ******** |           X
  9 |-.239*|        ***|            |*******|-.178*|         ** |           X
 10 |-.484*|      *****|            |*******|-.053 |          * |           X
 11 |-.486*|      *****|            |*******|-.030 |          * |           X
 12 |-.245*|        ***|            |*******|-.041 |          * |           X
 13 | .239*|           |***         |*******|-.068*|          * |           X
 14 | .962*|           |*********** |*******| .211*|            |**         X
 15 | .237*|           |***         |*******|-.166*|         ** |           X
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Time series identification for H9
PACF is computed using Yule-Walker equations.
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Lag |  Autocorrelation Function     |Box/Prc|    Partial Autocorrelations   X
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
  1 | .406*|           |****        |164.92*| .406*|            |****       X
  2 |-.040 |          *|            |166.49*|-.245*|        *** |           X
  3 |-.335*|       ****|            |278.42*|-.277*|        *** |           X
  4 |-.482*|      *****|            |511.04*|-.317*|        *** |           X
  5 |-.483*|      *****|            |744.21*|-.365*|       **** |           X
  6 |-.336*|       ****|            |857.15*|-.417*|      ***** |           X
  7 |-.042 |          *|            |858.90*|-.428*|      ***** |           X
  8 | .400*|           |****        |*******|-.166*|         ** |           X
  9 | .984*|           |*********** |*******| .973*|            |***********X
 10 | .396*|           |****        |*******|-.611*|    ******* |           X
 11 |-.046 |          *|            |*******|-.189*|         ** |           X
 12 |-.338*|       ****|            |*******|-.050 |          * |           X
 13 |-.484*|      *****|            |*******|-.010 |          * |           X
 14 |-.485*|      *****|            |*******|-.011 |          * |           X
 15 |-.339*|       ****|            |*******|-.035 |          * |           X
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
Descriptive Statistics
--------+---------------------------------------------------------------------
Variable|    Mean       Std.Dev.     Minimum      Maximum        Cases Missing
--------+---------------------------------------------------------------------
      H7|  .498364      .288477      .832986E-03  .997501         1000       0
      H9|  .497013      .288833      .152416E-03  .998628         1000       0
      R1|  .491032      .277905      .201170E-03  .995654         1000       0
      R2|  .502866      .282573      .943576E-04  .999873         1000       0
--------+---------------------------------------------------------------------
```

To illustrate the technique, we will use 500 observations from a simulated data set, and compare estimation with 500 random draws with estimation using 50 Halton draws. Computation time is linear in the number of draws, so the second model takes roughly one tenth as long as the first. (It is not exact as generation of a Halton draw does not take the same amount of time as generation of a random number.)

```
CALC        ; Ran(12345) $
SAMPLE      ; 1-2000 $
CREATE      ; I = Trn(10,0) $
MATRIX      ; b1 = Rndm(200)
            ; b2 = Rndm(200) $
CALC        ; b3 = -1 ; b4 = .5 ; b5 = .2 $
CREATE      ; x1 = Rnu(-.5,.5)
            ; x2 = Rnn(0,1)
            ; x3 = Rnn(0,1)
            ; x4 = Rnd(3)-2 $
CREATE      ; index = .5*b1(i) + .2*b2(i)*x1 + b3*x2 + b4*x3 + b5*x4 + Rnn(0,4) $
CREATE      ; y = index > 0 $
SAMPLE      ; 1-500 $
TIMER $
PROBIT      ; Lhs = y
            ; Rhs = one,x1,x2
            ; RPM
            ; Pts = 500
            ; Pds = 10
            ; Fcn = one(n),x1(n) $
PROBIT      ; Lhs = y
            ; Rhs = one,x1,x2
            ; RPM
            ; Pts = 50
            ; Pds = 10
            ; Halton
            ; Fcn = one(n),x1(n) $
```

The results are strikingly similar, but, the Halton method takes roughly one eighth of the time to complete the estimation. (The computation of the initial estimates is omitted for both models.)

```
-----------------------------------------------------------------------------
Random Coefficients  Probit   Model
Dependent variable                      Y
Log likelihood function      -336.40582
Restricted log likelihood    -336.41192
Chi squared [   2 d.f.]          .01218
Significance level               .99393
Sample is 10 pds and     50 individuals
PROBIT (normal)  probability model
Simulation based on 500 random draws
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Nonrandom parameters
      X2|    -.26254***      .05998    -4.38  .0000     -.38011   -.14497
        |Means for random parameters
Constant|     .05662         .05758      .98  .3254     -.05623    .16948
      X1|     .06019         .24729      .24  .8077     -.42449    .54488
        |Scale parameters for dists. of random parameters
Constant|     .03541         .05708      .62  .5350     -.07647    .14730
      X1|     .00757         .23486      .03  .9743     -.45274    .46789
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Elapsed time:     0 hours,  0 minutes,  6.02 seconds.
-----------------------------------------------------------------------------
Random Coefficients  Probit   Model
Dependent variable                      Y
Log likelihood function      -336.36654
Restricted log likelihood    -336.41192
Chi squared [   2 d.f.]          .09076
Significance level               .95564
McFadden Pseudo R-squared      .0001349
Estimation based on N =    500, K =   5
Inf.Cr.AIC  =  682.733 AIC/N =    1.365
Sample is 10 pds and     50 individuals
PROBIT (normal)  probability model
Simulation based on  50 Halton draws
--------+--------------------------------------------------------------------
        |                 Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Nonrandom parameters
      X2|    -.26320***      .06004    -4.38  .0000     -.38087   -.14553
        |Means for random parameters
Constant|     .05679         .05841      .97  .3309     -.05769    .17127
      X1|     .06042         .24702      .24  .8068     -.42374    .54458
        |Scale parameters for dists. of random parameters
Constant|     .07584         .05725     1.32  .1853     -.03638    .18805
      X1|     .02051         .23609      .09  .9308     -.44222    .48324
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
Elapsed time:     0 hours,  0 minutes,   .84 seconds.
```

Constructing the Parameter Vector

In the following, we include all cases by allowing for autocorrelation in the random effects. Thus, we write the random parameter vector as $\boldsymbol{\beta}_{it}$. Let

$\mathbf{v}_{itr}$ = the *r*th replication (random draw) on the vector of random variables $\mathbf{v}_{it}$

With $\mathbf{v}_{itr}$ in hand, we form the *r*th draw on the random parameter, $\boldsymbol{\beta}_{itr}$ as follows:

$\boldsymbol{\beta}_{1itr} = \boldsymbol{\beta}_1$ (K_1 nonrandom parameters – does not change with *i*, *r*, or *t*).
This parameter vector is being estimated.

$$\boldsymbol{\beta}_{2itr} = \boldsymbol{\beta}_2 + \boldsymbol{\Delta}\mathbf{z}_i + \boldsymbol{\Sigma}\mathbf{v}_{itr} + \boldsymbol{\Pi}\mathbf{v}_{itr} \quad (K_2 \text{random parameters})$$

$$= \boldsymbol{\beta}_2 + \boldsymbol{\Delta}\mathbf{z}_i + \boldsymbol{\Gamma}\mathbf{v}_{itr} \text{ where } \boldsymbol{\Gamma} = \boldsymbol{\Sigma} + \boldsymbol{\Pi}$$

where $\boldsymbol{\beta}_2$ = the fixed means of the distributions for the random parameters.
This parameter vector is being estimated.

$\mathbf{z}_i$ = a set of *M* observed variables which do not vary over time and which enter the means (optional)

$\boldsymbol{\Delta}$ = coefficient matrix, $K_2 \times M$, which forms the observation specific term in the mean. If $\mathbf{z}_i$ has been specified with **; Rpm = list**, then $\boldsymbol{\Delta}$ is a vector of parameters that is being estimated. Otherwise, $\boldsymbol{\Delta}$ does not appear in the model specification.

$\boldsymbol{\Sigma}$ = diagonal matrix of scale parameters – standard deviations if the random effects are uncorrelated ($\boldsymbol{\Pi} = \mathbf{0}$). $\boldsymbol{\Sigma}$ is being estimated.

$\boldsymbol{\Pi}$ = lower triangular matrix *with zeros on the main diagonal*. $\boldsymbol{\Pi}$ becomes nonzero only when you specify **; Correlation**. $\boldsymbol{\Pi}$ is a matrix of parameters to be estimated; otherwise, it does not appear in the model

$\boldsymbol{\Gamma}$ = lower triangular or diagonal matrix which produces the covariance matrix of the random parameters, $\boldsymbol{\Omega} = \boldsymbol{\Gamma}\boldsymbol{\Gamma}'$ in the random effects form and $\boldsymbol{\Omega} = \boldsymbol{\Gamma}(\mathbf{I}\text{-}\mathbf{R}^2)^{-1}\boldsymbol{\Gamma}'$ in the AR(1) model. The elements of $\boldsymbol{\Gamma}$ are being estimated.

$\mathbf{R}$ = Diagonal autocorrelation matrix. If **; AR1** is specified, then the elements of **R** are being estimated. Otherwise, **R** does not appear in the model

$\mathbf{x}_{1it}$ = variables multiplied by $\boldsymbol{\beta}_{1itr}$.

Various restrictions, such as zeros in $\boldsymbol{\Delta}$ or $\boldsymbol{\Gamma}$, or equality restrictions between means and variances, are imposed during the construction. The parameter vector, $\boldsymbol{\beta}_{itr}$ is now in hand. For parameters specified to be lognormally distributed, the respective element of $\boldsymbol{\beta}_{itr}$ is now exponentiated.

Forming the Simulated Likelihood

The probability density function is formed by beginning with

$$P_{itr} \quad = \; g(y_{it}, \boldsymbol{\beta}_{itr}{}'\mathbf{x}_{it}, \boldsymbol{\theta})$$

(Note, if this is the random effects model, then $\boldsymbol{\beta}_{itr}{}'\mathbf{x}_{it} = \boldsymbol{\beta}_{ir}{}'\mathbf{x}_{it}$.) The joint conditional probability for the ith individual is

$$P_{ir}|(\mathbf{v}_{itr}, t = 1,...,T_i) \; = \; \prod_{t=1}^{T_i} \; P_{itr}|\,\mathbf{v}_{itr}.$$

The unconditional density would now be obtained by integrating the random terms out of the conditional distribution. We do this by simulation:

$$P_i \quad = \; \frac{1}{R}\sum_{r=1}^{R} \; P_{ir}|(\mathbf{v}_{itr}, t = 1,...,T_i)$$

Note that in the random effects case, we are averaging over R replications in which the T_i observations are each a function of the same $\mathbf{v}_{ir}$. Thus, each replication in this case involves drawing a single random vector. In the AR1 case, each replication involves drawing a sequence of T_i vectors, $\mathbf{v}_{itr}$. Finally, the simulated log likelihood function to be maximized is

$$\begin{aligned}\log L \quad &= \; \sum_{i=1}^{N} \; \log P_i \\ &= \; \sum_{i=1}^{N} \; \log \frac{1}{R}\sum_{r=1}^{R} \; P_{ir}\,|(\mathbf{v}_{itr}, t = 1,...,T_i) \\ &= \; \sum_{i=1}^{N} \; \log \; \frac{1}{R}\sum_{r=1}^{R} \; \prod_{t=1}^{T_i} \; P_{itr}|\,\mathbf{v}_{itr}\end{aligned}$$

Derivatives of the Simulated Log Likelihood

The derivatives of the log likelihood function must be approximated as well. The theoretical maximum is based on

$$\frac{\partial \log L_i}{\partial \boldsymbol{\alpha}} = \frac{1}{L_i}\frac{\partial L_i}{\partial \boldsymbol{\alpha}}$$

where $\boldsymbol{\alpha}$ is the vector of all parameters in the model. Then,

$$\frac{\partial L_i}{\partial \boldsymbol{\alpha}} \quad = \quad \int_{\text{Range of } \mathbf{v}_{it}} g(\mathbf{v}_{it}, t=1,...,T_i)\frac{\partial}{\partial \boldsymbol{\alpha}}\prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}_{it}'\mathbf{x}_{it}, \boldsymbol{\theta}) \; d(\mathbf{v}_{it}, t=1,...,T_i)$$

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\alpha}}\prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}_i'\mathbf{x}_{it}, \boldsymbol{\theta}) \quad &= \quad \sum_{t=1}^{T_i}\left[\frac{\partial P(y_{it}, \boldsymbol{\beta}_i'\mathbf{x}_{it}, \boldsymbol{\theta})}{\partial \boldsymbol{\alpha}}\right]\prod_{s\neq t} P(y_{is}, \boldsymbol{\beta}_i'\mathbf{x}_{it}, \boldsymbol{\theta}) \\ &= \quad \prod_{t=1}^{T_i} P(y_{it}, \boldsymbol{\beta}_{it}'\mathbf{x}_{it}, \boldsymbol{\theta})\sum_{t=1}^{T_i}\left[\frac{\partial \log P(y_{it}, \boldsymbol{\beta}_{it}'\mathbf{x}_{it}, \boldsymbol{\theta})}{\partial \boldsymbol{\alpha}}\right]\end{aligned}$$

Collecting terms once again, we obtain the approximation,

$$\frac{\partial \log L}{\partial \boldsymbol{\alpha}} = \sum_{i=1}^{N} \frac{1}{L_i}\frac{\partial L_i}{\partial \boldsymbol{\alpha}}$$

$$\approx \sum_{i=1}^{N} \frac{\left\{ \frac{1}{R}\sum_{r=1}^{R}\left[\prod_{t=1}^{T_i} P(y_{it},\boldsymbol{\beta}_{itr}'\mathbf{x}_{it},\boldsymbol{\theta})\right]\left[\sum_{t=1}^{T_i}\frac{\partial \log P(y_{it},\boldsymbol{\beta}_{itr}'\mathbf{x}_{it},\boldsymbol{\theta})}{\partial \boldsymbol{\alpha}}\right]\right\}}{\left\{\frac{1}{R}\sum_{h=1}^{H}\left[\prod_{t=1}^{T_i} P(y_{it},\boldsymbol{\beta}_{itr}'\mathbf{x}_{it},\boldsymbol{\theta})\right]\right\}}$$

We now consider in finer detail how the derivatives with respect to the low level, structural parameters are computed.

To avoid some involved notation, it is useful to go back and partition the parameter vector. Each element of $\boldsymbol{\beta}_{itr}$ involves its own vector of structural parameters, a β_k from the mean vector, a row of $\boldsymbol{\Delta}$, a diagonal element of $\boldsymbol{\Sigma}$, a row of $\boldsymbol{\Pi}$, and an autocorrelation coefficient, ρ_k. It is convenient to write this explicitly as

$$\beta_{k,itr} = \beta_k + \boldsymbol{\delta}_k'\mathbf{z}_i + \sigma_k v_{k,itr} + \boldsymbol{\pi}_k'\mathbf{v}_{itr}$$

$$\mathbf{v}_{k,itr} = \rho_k v_{\mathbf{k,i},t-1,r} + u_{k,itr} \text{ or } \mathbf{v}_{k,1tr} = u_{k,i1r}/\sqrt{1-\rho_k^2} \text{ for the first observation.}$$

Note that the first row of $\boldsymbol{\Pi}$ has no nonzero elements, the second has only one, the third, two, and so on. Also, for any nonrandom parameters, all terms save the first are zero in the statement above. It is useful, as well, to keep in mind that $\boldsymbol{\Gamma}$ is a triangular matrix, so the vector $\boldsymbol{\gamma}_k$ has one, then two, then ... nonzero elements. Let $\boldsymbol{\mu}_k$ denote the full vector of parameters associated with $\beta_{k,itr}$, and collect these subvectors in a large, complete vector of parameters, $\boldsymbol{\mu}$. (At this point, $\boldsymbol{\mu}$ is the full vector of model parameters except for any ancillary parameters such as θ in the negative binomial or the disturbance standard deviation, σ, in the tobit model.) We seek the gradient,

$$\frac{\partial \log L}{\partial \boldsymbol{\mu}} = \sum_{i=1}^{N} \frac{\partial \log P_i}{\partial \boldsymbol{\mu}}$$

which we will obtain by stacking the gradients with respect to $\boldsymbol{\mu}_k$. For the moment, assume that the random parameters are uncorrelated, so that the matrix, $\boldsymbol{\Pi}$ is not in the model – the reason is that with nonzero $\boldsymbol{\Pi}$ and autocorrelation, each β_k is a function of the ρ_m in other parameters. We turn to this complication below. At various points, we will use analogs to the convenient result

$$\frac{\partial P}{\partial \theta} = P\frac{\partial \log P}{\partial \theta}.$$

Suppressing the conditioning notation, and collecting terms from above, we have

$$\frac{\partial \log P_i}{\partial \boldsymbol{\mu}_k} = \frac{1}{P_i}\frac{1}{R}\sum_{r=1}^{R} P_{ir}\frac{\partial \log P_{ir}}{\partial \boldsymbol{\mu}_k} = \frac{1}{P_i}\frac{1}{R}\sum_{r=1}^{R} P_{ir}\sum_{t=1}^{T_i}\frac{\partial \log P_{itr}}{\partial \boldsymbol{\mu}_k}$$

The second expression takes us into the primitive, lowest level probabilities. To evaluate them, we can now use the simplification

$$\frac{\partial \log P_{itr}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \log P_{itr}}{\partial \beta_{k,itr}} \frac{\partial \beta_{k,itr}}{\partial \boldsymbol{\mu}_k}.$$

The first term in the product is the familiar one from maximization of the conventional log likelihoods for these models. For example, for the Poisson model,

$$\frac{\partial \log P_{itr}}{\partial \beta_{k,itr}} = (y_{it} - \lambda_{itr})x_{k,it}$$

while for the negative binomial model, this is

$$\frac{\partial \log P_{itr}}{\partial \beta_{k,itr}} = [\theta(y_{it} + q_{itr}) - \theta]x_{k,it}.$$

In the absence of cross parameter correlation, the latter derivatives are easy to evaluate:

$$\frac{\partial \beta_{k,itr}}{\partial \beta_k} = 1, \quad \frac{\partial \beta_{k,itr}}{\partial \boldsymbol{\delta}_k} = \mathbf{z}_i, \quad \frac{\partial \beta_{k,itr}}{\partial \sigma_k} = v_{k,itr} \text{, and, if } \mathbf{; Cor}, \ \frac{\partial \beta_{k,itr}}{\partial \boldsymbol{\pi}_k} = \mathbf{v}_{itr}$$

Note that the last of these has only k-1 nonzero terms, for k = 1,...,K_2. Finally, for the autocorrelation parameters (assuming that the parameters are not correlated),

$$\frac{\partial \beta_{k,itr}}{\partial \rho_k} = \sigma_k \frac{\partial v_{k,itr}}{\partial \rho_k} = \sigma_k v_{k,i,t-1,r}$$

or $$\sigma_k u_{k,i1r} \frac{\rho_k}{(1-\rho_k^{\,2})^{3/2}}$$

for the first observation.

One term remains. If **; Cor** has been specified, then β_k is a function of ρ_m for some of the other parameters, because the term $\boldsymbol{\pi}_k'\mathbf{v}_{itr}$ is a function of some alien components of $\mathbf{v}_{itr}$ (and, by construction, it is not a function of the kth component). In particular,

$$\frac{\partial \log P_i}{\partial \rho_k} = \frac{1}{P_i}\frac{1}{R}\sum\nolimits_{r=1}^{R} P_{ir}\frac{\partial \log P_{ir}}{\partial \rho_k} = \frac{1}{P_i}\frac{1}{R}\sum\nolimits_{r=1}^{R} P_{ir}\sum\nolimits_{t=1}^{T_i}\frac{\partial \log P_{itr}}{\partial \rho_k}$$

and

$$\frac{\partial \log P_{itr}}{\partial \rho_k} = \frac{\partial \log P_{itr}}{\partial \beta_{k,itr}}\frac{\partial \beta_{k,itr}}{\partial \rho_k} + \sum\nolimits_{m \neq k} \frac{\partial \log P_{itr}}{\partial \beta_{m,itr}}\frac{\partial \beta_{m,itr}}{\partial \rho_k}$$

The final term is

$$\frac{\partial \beta_{m,itr}}{\partial \rho_k} = \pi_{mk}\frac{\partial v_{k,itr}}{\partial \rho_k}\ .$$

The last derivative above was given previously. Working backwards from here, we assemble these parts to obtain the full gradient of the log likelihood.

The common ancillary parameters are a remaining complication. Consider, for example, the negative binomial model. For the overdispersion parameter,

$$\frac{\partial \log P_i}{\partial \theta} = \frac{1}{P_i}\frac{1}{R}\sum\nolimits_{r=1}^{R} P_{ir}\sum\nolimits_{t=1}^{T_i}\frac{\partial \log P_{itr}}{\partial \theta}$$

and the latter derivative is

$$\frac{\partial \log P_{itr}}{\partial \theta} = \Psi(\theta + y_{it}) - \Psi(\theta) + \log q_{itr} + (1 - q_{itr}) - y_{iit}q_{itr}/\theta.$$

Another formulation of the preceding computation is illuminating. This shows more graphically how the preceding are actually carried out – the implementation is actually simpler than the derivation might suggest. We produce this for the group of binary choice models as an illustration. For the binary choice models, we have

$$P(y_{it}, \boldsymbol{\beta}_{itr}'\mathbf{x}_{it}) = F[\boldsymbol{\beta}_{itr}'\mathbf{x}_{it}]^{y_{it}}\left(1 - F[\boldsymbol{\beta}_{itr}'\mathbf{x}_{it}]\right)^{1-y_{it}} = G_{itr}$$

so

$$\log P(y_{it},\boldsymbol{\beta}_{itr}'\mathbf{x}_{it}) = y_{it}\log F_{itr} + (1 - y_{it})\log(1 - F_{itr}).$$

The index is

$$\begin{aligned} w_{itr} &= \boldsymbol{\beta}_{itr}'\mathbf{x}_{it} \\ &= \boldsymbol{\beta}'\mathbf{x}_{it} + \mathbf{z}_i'\boldsymbol{\Delta}'\mathbf{x}_{it} + \mathbf{v}_{itr}'\boldsymbol{\Gamma}'\mathbf{x}_{it} \end{aligned}$$

We will need

$$\frac{\partial w_{irt}}{\partial \boldsymbol{\mu}} = \begin{bmatrix} \mathbf{x}_{it} \\ \mathbf{z}_i \otimes \mathbf{x}_{it} \\ \mathbf{v}_{ir} \otimes \mathbf{x}_{it} \end{bmatrix} = \mathbf{h}_{itt}$$

Then,

$$\frac{\partial \log P(y_{it}, w_{irt})}{\partial \boldsymbol{\mu}} = \left(\frac{y_{it}}{F_{irt}} - \frac{1-y_{it}}{1-F_{irt}}\right) g_{itr}(y_{it}, w_{itr})\mathbf{h}_{itr} = \mathbf{g}_{itr}.$$

The forms of the particular density functions, $g_{it}(.)$, differ among the four models. The functional forms appear at the beginning of Section E27.2.1. In the vector at the end of the expression, the lower term is the result of the term $\mathbf{x}_{it}'\boldsymbol{\Gamma}\mathbf{v}_{ir}$. Since $\boldsymbol{\Gamma}$ is a lower triangular matrix, this term actually involves the $K_2(K_2+1)/2$ terms that are nonzero in the matrix $\boldsymbol{\Gamma}$including σ_k and nonzero elements of $\boldsymbol{\Pi}$.

Hessians and Asymptotic Covariance Matrix Estimation

We will only sketch the full derivation of the Hessians here. Return to the full gradient of the *i*th term in the log likelihood log likelihood – terms are summed over *i* to get the gradient and Hessian – the following is written in terms of the full parameter vector, including any ancillary parameters. The gradient is

$$\mathbf{g}_i = \frac{1}{P_i}\frac{1}{R}\sum_{r=1}^{R} P_{ir}\sum_{t=1}^{T_i} \frac{\partial \log P_{itr}}{\partial \boldsymbol{\alpha}}.$$

Let $\mathbf{H}_i$ denote the second derivatives matrix. Then,

$$\begin{aligned}\mathbf{H}_i = - \mathbf{g}_i\, \mathbf{g}_i' \quad & + \frac{1}{P_i}\frac{1}{R}\sum_{r=1}^{R} P_{ir}\left(\sum_{t=1}^{T_i} \frac{\partial \log P_{itr}}{\partial \boldsymbol{\alpha}}\right)\left(\sum_{t=1}^{T_i} \frac{\partial \log P_{itr}}{\partial \boldsymbol{\alpha}}\right)' \\ & + \frac{1}{P_i}\frac{1}{R}\sum_{r=1}^{R} P_{ir}\sum_{t=1}^{T_i} \frac{\partial^2 \log P_{itr}}{\partial \boldsymbol{\alpha}\, \partial \boldsymbol{\alpha}'}.\end{aligned}$$

The only term which has not already appeared is the second derivatives matrix in the third part. Consider first the case of no autocorrelation. This derivative is obtained by differentiation of

$$\frac{\partial \log P_{itr}}{\partial \boldsymbol{\mu}_k} = \frac{\partial \log P_{itr}}{\partial \beta_{k,itr}}\frac{\partial \beta_{k,itr}}{\partial \boldsymbol{\mu}_k}.$$

which gives

$$\frac{\partial^2 \log P_{itr}}{\partial \boldsymbol{\mu}_k \partial \boldsymbol{\mu}_m'} = \frac{\partial \log P_{itr}}{\partial \beta_{k,itr}}\frac{\partial^2 \beta_{k,itr}}{\partial \boldsymbol{\mu}_k \partial \boldsymbol{\mu}_m'} + \frac{\partial \beta_{k,itr}}{\partial \boldsymbol{\mu}_k}\frac{\partial^2 \log P_{itr}}{\partial \beta_{k,itr} \partial \boldsymbol{\mu}_m'}.$$

In the absence of autocorrelation, the random parameters are linear in the underlying parameters, so the first of the two second derivatives is zero. Using this and our previous results, we obtain

$$\frac{\partial^2 \log P_{itr}}{\partial \boldsymbol{\mu}_k \partial \boldsymbol{\mu}_m'} = \left(\frac{\partial^2 \log P_{itr}}{\partial \beta_{k,itr} \partial \beta_{m,itr}}\right)\left(\frac{\partial \beta_{k,itr}}{\partial \boldsymbol{\mu}_k}\right)\left(\frac{\partial \beta_{m,itr}}{\partial \boldsymbol{\mu}_m'}\right).$$

The only major complication in the preceding arises when there is autocorrelation, as in this case, and when the reduced form parameters are not linear in ρ_k. In this instance, the square of the first derivative is used as an approximation to the second. The BFGS algorithm is always used for estimation of this model, so the Hessian is only used at exit for computing the asymptotic covariance matrix.

R24.8 Multilevel and Multiple Effects RP Models

The following applies to all random parameters models in *LIMDEP* – the entire class of models estimable with the **; RPM** specification with only the exception of the two equation models, bivariate probit and sample selection. It modifies the single index models.

The model is based on the single index function

$$Index_{it} = \boldsymbol{\beta}'\mathbf{x}_{it}$$

such as the linear regression model, $y_{it} = Index_{it} + \varepsilon_{it}$ or the probit model, in which $y_{it} = 1(Index_{it}+\varepsilon_{it}> 0)$ or the Poisson regression in which the probabilities are functions of $\exp(Index_{it})$. We add to this M = up to 10 'effects'

$$Index_{it} = \boldsymbol{\beta}'\mathbf{x}_{it} + c_{j1}\omega_{j1,i} + c_{j2}\omega_{j2,i} + \ldots\, c_{jM}\,\omega_{jM,i}.$$

The c_{jM} are ones and zeros simply used to select the effects in the model. The effects are up to 10 normally distributed random terms associated with discrete group indicators such as strata, clusters, etc. Effects may appear singly or as products, and may be nested or simply be associated with any desired groupings of the data. The associated variables can be any desired discrete indicator that associates a unique value with a group. Consider an example based on test scores. Suppose we have nationwide data, arranged by *region*, *state*, *county*, *district*, *school*. That is individual test scores observed in five decreasing levels of aggregation. Then, in addition to the data on test scores (presumably individual students) and the covariates in **x**, we have variables with distinct codes for the five levels of aggregation – the only restriction is that codes must be integers from 1,2,...,9999. The specification is

; REM = name1, name2, ..., nameM

For our example, this would thus be

; REM = region,state,county,district,school

This estimator does not require that these 'effects' be nested. The effects can be defined at any level of aggregation, and could be a mixture of nested and nonnested groupings. Suppose, for example, you also had indicators of grouping by *type of program*, which might be one of, say, 10, which varies all over the range of observations, without respect to the other five groupings listed. For another example, one might also have a *party* effect in that list, for whether the state in question had a *Democratic*, *Republican*, or *Other Party* governor at the time. This could also be included.

Effects may also be main or secondary (products). You can specify secondary effects by writing the effects as products, as in

; REM = name1, name2, name3*name4, name2*name3*name4, name1*name4

You may define up to 10 effects or combinations of effects in total, using up to 10 basic effects. To continue the example, you might specify an interaction between state and district with

; REM = region,state,county,district,school,state*district

; REM can be added to an RPM model or may appear by itself instead of **RPM ; Fcn = …**

R24.8.1 Command

This estimator uses *LIMDEP*'s package of random parameter model (RPM) estimators. The essential part of the command is

```
Model   ; Lhs  = the dependent variable
        ; Rhs  = the independent variables
        ; RPM
        ; Pds  = the correct specification for your panel (see below)
        ; REM  = the specification of your random effects  $
```

This specification may be in addition to other random parameter specifications in the command. (See the application below.)

Typically, the panel specification in **; Pds = ...** would correspond to the structure of one of your effects variables. But, this is not required. Indeed, you could have **; Pds = 1**. But, if you are analyzing a panel, you should specify it as usual. Note that the command does not contain **; Panel**. This must be omitted from this command. The effects are set up as described above. There is one other specification that you should use. The estimator for this model is maximum simulated likelihood. You may want to control the number of random draws used in the simulations. This is an extremely computation intensive estimator. The number of random draws is specified with

```
        ; Pts = the desired number
```

The default value is 100. For generating final results in a study, you will probably use several hundred. But, for exploratory work, as in our example below, you might want to choose a small value, such as 10 or 25. Also, as in the case of the RPM, you may gain some speed and smoothness of the optimization by using Halton sequences instead of pseudorandom draws to do the integrations involved. This is done by adding

```
        ; Halton
```

to the command.

R24.8.2 Application

We consider a random parameters stochastic frontier model (see Chapter E64) of the form

$$y_{it} = \alpha + \boldsymbol{\beta}_i'\mathbf{x}_{it} + \varepsilon_{it} - a_{it} + \sigma_u u_i + \sigma_w w_g + \gamma(\sigma_u u_i)(\sigma_v w_g)$$

The parameter vector is treated as a simple RPM, with $\beta_{i,k} = \beta_k^0 + v_{i,k}$. In the setting of the frontier model, a_{it} is the half normally distributed inefficiency term while ε_{it} (normally denoted v_{it}) is the symmetric noise term. Note that in this model, we allow these to vary with time. The common effect $\sigma_u u_i$ is equivalent to a producer specific effect, which is, in turn, also equivalent to a random constant term. In the application below, we fit the model to a sample of farms in a panel data set with $T = 6$ observations per farm. (The data are discussed in greater detail in Section E19.9.2.) We are also artificially (for the purpose of our numerical example) grouping farms into clusters of six farms. (There are 247 farms, so the last cluster has only one farm.) We allow for separate effects for farm and group, and also provide for an interaction of the two effects.

The commands are

```
CREATE      ; group = Trn(36,0) $
FRONTIER    ; Lhs = yit ; Rhs = one,x1,x2,x3,x4 $
FRONTIER    ; Lhs = yit ; Rhs = one,x1,x2,x3,x4
            ; RPM  ; Fcn = x1(n),x2(n),x3(n),x4(n)
            ; Pds = 6
            ; Pts = 15
            ; REM = group,farm $
FRONTIER    ; Lhs = yit ; Rhs = one,x1,x2,x3,x4 $
FRONTIER    ; Lhs = yit ; Rhs = one,x1,x2,x3,x4
            ; RPM ; Fcn = x1(n),x2(n),x3(n),x4(n)
            ; Pds = 6
            ; Pts = 15
            ; REM  = group,farm,group*farm $
```

Three sets of estimates appear below. The first is the pooled, fixed parameter version of the model that is used to provide the starting values for the iterations. The second is the RP model with farm and group effects. The third adds the interaction effect of farm and group. Since the grouping is completely artificial, we would not expect to see any significant effect due to this component. Although the coefficient on the effects in the two RE models do appear to be significant, the variances due to the grouping effects are very small – one and two orders of magnitude smaller – compared to the farm effect which can be clearly seen in any panel data model fit with these two data. We might surmise that the 'group effect' here is picking up farm effects that are left over beyond the simple linear additive effect of the first random effect. Note, finally, in the third model, the iterative routine exited abnormally with a diagnostic that no improvement in the function could be located. However, this was after 40 iterations, and in fact, the derivatives were quite small at this point. This suggests only that the likelihood is fairly flat at this point, not that the optimization has failed, and we take the estimates as given.

```
-----------------------------------------------------------------------------
Limited Dependent Variable Model - FRONTIER
Dependent variable                      YIT
Log likelihood function           822.68831
Estimation based on N =   1482, K =   7
Inf.Cr.AIC  =-1631.377 AIC/N =   -1.101
Variances: Sigma-squared(v)=     .01075
           Sigma-squared(u)=     .02425
           Sigma(v)        =     .10371
           Sigma(u)        =     .15573
Sigma = Sqr[(s^2(u)+s^2(v)]=     .18710
Stochastic Production Frontier, e = v-u
```

```
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Primary Index Equation for Model
Constant|    11.7014***      .00447  2614.87  .0000     11.6926   11.7101
      X1|     .58369***      .01887    30.93  .0000      .54670    .62068
      X2|     .03555***      .01113     3.20  .0014      .01375    .05736
      X3|     .02256*        .01281     1.76  .0783     -.00256    .04768
      X4|     .44948***      .01035    43.42  .0000      .42919    .46977
        |Variance parameters for compound error
  Lambda|    1.50164***      .08748    17.17  .0000     1.33019   1.67310
   Sigma|     .18710***      .00011  1698.90  .0000      .18688    .18732
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
Random Coefficients  Frontier Model
Dependent variable                  YIT
Log likelihood function      1266.54960
Restricted log likelihood        .00000
Chi squared [   6 d.f.]      2533.09921
Significance level               .00000
Estimation based on N =   1482, K =  13
Inf.Cr.AIC  =-2507.099 AIC/N =   -1.692
Sample is  6 pds and    247 individuals
Stochastic frontier (half normal)
Simulation based on  15 random draws
Model contained  2 random effects.
Sigma( u) (1 sided)  =      .08573
Sigma( v) (symmetric)=      .06173
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Production / Cost parameters, nonrandom first
Constant|    11.6209***      .00389  2988.25  .0000     11.6133   11.6285
        |Means for random parameters
      X1|     .63225***      .01001    63.17  .0000      .61264    .65187
      X2|    -.00627         .00606    -1.03  .3016     -.01815    .00562
      X3|     .05448***      .00706     7.72  .0000      .04064    .06831
      X4|     .41693***      .00537    77.63  .0000      .40641    .42746
        |Scale parameters for dists. of random parameters
      X1|     .03083***      .00398     7.75  .0000      .02303    .03863
      X2|     .02501***      .00424     5.89  .0000      .01669    .03332
      X3|     .03725***      .00554     6.73  .0000      .02640    .04810
      X4|     .07809***      .00280    27.89  .0000      .07260    .08358
        |Standard Deviations of Random Effects
R.E.(01)|     .05930***      .00218    27.14  .0000      .05502    .06358
R.E.(02)|     .15460***      .00259    59.74  .0000      .14953    .15968
        |Variance parameter for v +/- u
   Sigma|     .10565***      .00270    39.13  .0000      .10036    .11094
        |Asymmetry parameter, lambda
  Lambda|    1.38874***      .10644    13.05  .0000     1.18013   1.59735
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
+---------------------------------------------+--------------+
| Random effects in the model are based on    |Random Effect |
| these expanded qualitative variables.       |     Variance |
| R.E.(01) = GROUP                            |      .003516 |
| R.E.(02) = FARM                             |      .023903 |
+---------------------------------------------+--------------+
```

```
-----------------------------------------------------------------------------
Random Coefficients  Frontier Model
Dependent variable                  YIT
Log likelihood function      1278.51141
Restricted log likelihood        .00000
Chi squared [   7 d.f.]      2557.02282
Significance level               .00000
Estimation based on N =   1482, K =  14
Inf.Cr.AIC  =-2529.023 AIC/N =   -1.706
Sample is  6 pds and    247 individuals
Stochastic frontier (half normal)
Simulation based on  15 random draws
Model contained  3 random effects.
Sigma( u) (1 sided)  =      .08949
Sigma( v) (symmetric)=      .06025
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
     YIT|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Production / Cost parameters, nonrandom first
Constant|    11.6173***      .00397  2929.12  .0000     11.6095   11.6251
        |Means for random parameters
      X1|     .65045***      .01033    62.98  .0000      .63021    .67069
      X2|     .01572**       .00623     2.52  .0116      .00352    .02793
      X3|     .02458***      .00738     3.33  .0009      .01011    .03905
      X4|     .40630***      .00553    73.41  .0000      .39545    .41714
        |Scale parameters for dists. of random parameters
      X1|     .01456***      .00421     3.45  .0006      .00629    .02282
      X2|     .06401***      .00501    12.78  .0000      .05420    .07383
      X3|     .10801***      .00623    17.34  .0000      .09580    .12022
      X4|     .07088***      .00271    26.13  .0000      .06557    .07620
        |Standard Deviations of Random Effects
R.E.(01)|     .04544***      .00195    23.30  .0000      .04162    .04926
R.E.(02)|     .12743***      .00239    53.21  .0000      .12273    .13212
R.E.(03)|     .01908***      .00193     9.87  .0000      .01529    .02287
        |Variance parameter for v +/- u
   Sigma|     .10789***      .00271    39.87  .0000      .10258    .11319
        |Asymmetry parameter, lambda
  Lambda|    1.48523***      .11417    13.01  .0000     1.26147   1.70899
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

```
+---------------------------------------------+--------------+
| Random effects in the model are based on    |Random Effect |
| these expanded qualitative variables.       |     Variance |
| R.E.(01) = GROUP                            |      .002065 |
| R.E.(02) = FARM                             |      .016237 |
| R.E.(03) = GROUP    FARM                    |      .000364 |
+---------------------------------------------+--------------+
```

R24.8.3 Technical Details

The probability density function for the random parameters model, now with the additional effects, is formed by beginning with

$$P_{itr} \quad = \quad g(y_{it}, \boldsymbol{\beta}_{itr}'\mathbf{x}_{it} + c_{j1}\omega_{j1,i} + c_{j2}\omega_{j2,i} + \ldots c_{jM}\,\omega_{jM,i}, \boldsymbol{\theta})$$

The joint conditional probability for the *i*th individual is

$$P_{ir}|(\omega_{j1,i},\omega_{j2,i},\ldots\omega_{jM,i},\mathbf{v}_{itr},\, t = 1,\ldots,T_i) \;=\; \prod_{t=1}^{T_i} P_{itr}|\omega_{j1,i},\omega_{j2,i},\ldots\omega_{jM,i},\mathbf{v}_{itr}.$$

The unconditional density would now be obtained by integrating the random terms out of the conditional distribution. We do this by simulation, as we did before:

$$P_i \quad = \quad \frac{1}{R}\sum_{r=1}^{R} P_{ir}|(\omega_{j1,ir},\omega_{j2,ir},\ldots\omega_{jM,ir},\mathbf{v}_{itr},\, t = 1,\ldots,T_i)$$

Thus, each replication in this case involves drawing a single random vector on $\mathbf{v}_i$, but in addition, it involves drawing from the population of the common effects, $\omega_{j1,i},\omega_{j2,i},\ldots\omega_{jM,i}$. Finally, the simulated log likelihood function to be maximized is

$$\begin{aligned}
\log L &= \sum_{i=1}^{N} \log P_i \\
&= \sum_{i=1}^{N} \log \frac{1}{R}\sum_{r=1}^{R} P_{ir}\,|(\omega_{j1,i},\omega_{j2,i},\ldots\omega_{jM,i},\mathbf{v}_{itr},\, t = 1,\ldots,T_i) \\
&= \sum_{i=1}^{N} \log \frac{1}{R}\sum_{r=1}^{R} \prod_{t=1}^{T_i} P_{itr}|\omega_{j1,i},\omega_{j2,i},\ldots\omega_{jM,i},\mathbf{v}_{itr}
\end{aligned}$$

The procedure that includes these random effects is essentially the same save for the drawing of the effects for the simulations, which is not done in synchronization with the draws for the random parameters, but rather, is drawn from a separate reservoir of draws for the effects, themselves. Further details on the maximization appear above in Section R24.7.

We note one important aspect of the simulation/integration. Where the common effect is of the form $\sigma_\omega u_i$ – that is, the subscript on the effect matches the index of the product operation, as in the familiar random effects model – then the preceding is exactly equivalent to that RE model. In other cases, however, the effect may be varying over a different range than the index in the product. Consider a model with both group and time effects. There are T time effects for each i, since each individual is observed in each period. Thus,

$$Index_{it,r} = \boldsymbol{\beta}_{it,r}'\mathbf{x}_{it} + \gamma_1 v_{i,r} + \gamma_2 w_{t,r} + \gamma_3 v_{i,r} w_{t,r}.$$

That is, the integral over periods is recomputed for each i, while the integral over v_i is only computed once. Moreover, in principle, though w_t is a 'time' effect, we are treating it as if it were a state specific time effect when we integrate it out. (There is a separate random variable w_t for each period, however.) This means that although state observations are correlated across states because of the common time effect, we are treating them as uncorrelated by this procedure. Thus, it must be treated as approximate. This all comes out appropriately if the effects are nested and T_i corresponds to the highest level in the nest (the level that encompasses the lower levels).

R25: Latent Class Models

R25.1 Latent Class Models

A model for a panel of data, $i = 1,...,N$, $t = 1,...,T_i$ is specified with

$$P[y_{it}|\, \mathbf{x}_{it}] = F(y_{it},\boldsymbol{\beta}'\mathbf{x}_{it}) = P(i,t).$$

We use the term 'group' to indicate the T_i observations on respondent i in periods $t = 1,...,T_i$. We emphasize, *the latent class, or finite mixture model, does not require panel data, and has been widely analyzed in many settings using cross section data*. Indeed, the original finite mixture problem was cast in the context of cross section data. Thus, throughout this discussion, T_i may equal one for some or all i.

Unobserved heterogeneity in the distribution of y_{it} is assumed to impact the density in the form of a random effect. The continuous distribution of the heterogeneity is approximated by using a finite number of 'points of support.' The distribution is approximated by estimating the location of the support points and the mass (probability) in each interval. In implementation, it is convenient and useful to interpret this discrete approximation as producing a sorting of individuals (by heterogeneity) into J classes, $j = 1,...,J$. (Since this is an approximation, J is chosen by the analyst.)

Thus, we modify the model for a latent sorting of y_{it} into J 'classes' with a model which allows for heterogeneity as follows: The probability of observing y_{it} given that regime j applies is

$$P(i,t|j) = P[y_{it}|\, \mathbf{x}_{it}, j]$$

where the density is now specific to the group. The analyst does not observe directly which class, $j = 1,...,J$ generated observation $y_{it}|j$, and class membership must be estimated. Heckman and Singer (1984) suggest a simple form of the class variation in which only the constant term varies across the 3classes. This would produce the model

$$P(i,t|j) = F[y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta_j], \text{Prob}[\text{class} = j] = F_j$$

We might formulate this approximation more generally as,

$$P(i,t|j) = F[y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \boldsymbol{\delta}_j'\mathbf{x}_{it}],$$

$$F_j = \exp(\theta_j) / \Sigma_j \exp(\theta_j), \text{ with } \theta_J = 0.$$

In this formulation, each group has its own parameter vector, $\boldsymbol{\beta}_j' = \boldsymbol{\beta} + \boldsymbol{\delta}_j$, though the variables that enter the mean are assumed to be the same. (This can be changed by imposing restrictions on the full parameter vector, as described below.) This allows the Heckman and Singer formulation as a special case by imposing restrictions on the parameters. A further generalization is discussed below. In general, the latent class model will extend beyond the basic index function formulation.

There is a huge literature on latent class modeling. One particularly useful reference is a compendium by McLachlan and Peel (2000).

R25.2 Commands for Latent Class Modeling

The estimation command for this model is

```
Model, such as POISSON, PROBIT, TOBIT, LOGIT, etc.
                  ; Lhs  = dependent variable
                  ; Rhs  = independent variables
                  ; LCM  (for latent class model)
                  ; Pds  = panel data specification (optional)
                  ; Pts  = the number of classes  $
```

The estimator does not require a panel. You can use **; Pds = 1**. (This is the classic 'finite mixture' problem, which was not originally specified for panel data.) This applies to both *LIMDEP*'s class of estimators and to the LCM model in *NLOGIT*. The default number of support points is five. You may set *J* to 2, 3, ..., 9 with

; Pts = the value you wish.

Some particular results computed for the latent class model are

; Parameters	to save the individual specific parameter estimates. See Section R25.7 for details.
; Group = name	to retain the index of the most likely latent class
; Cprob = name	to retain the estimated probability for the most likely latent class

You can obtain a listing of the predicted class and the probabilities by using

; List

An example appears below. (Computation of these values is described in the technical details.) You can use the **; Rst = list** option to structure the latent class model so that different variables appear in different classes. Alternatively, you can use this to force the Heckman and Singer form of the model as follows, where we use a three class model as an example:

```
NAMELIST     ; x = ... one, list of variables $
CALC         ; k1 =  Col(x) - 1 $
POISSON      ; Lhs  = ...
             ; Rhs  = x
             ; LCM
             ; Pts = 3
             ; Rst = d1,k1_b, d2,k1_b, d3,k1_b, t1,t2,t3 $
```

This sequence of instructions computes a three class Poisson regression model in which the slope parameters are the same in all three classes – only the constant terms differ across the classes. Latent class models can be specified as an alternative to random effects models or random parameters models. The **; Rst = list** specification can be used to impose a variety of kinds of restrictions in the model. Note that the last three parameters in the list are for the class probabilities. *LIMDEP*, itself, imposes a restriction on these – the last parameter is fixed at zero. But, although you must include specifications for these parameters in your **; Rst** list, you should not impose any restrictions yourself on these parameters. They must remain unrestricted for the estimator to manipulate internally.

R25.3 Modeling Frameworks for Latent Class Analysis

The following modeling frameworks may be analyzed in the latent class structure:

- Linear regression, $y_{it}|j = \boldsymbol{\beta}_j'\mathbf{x}_{it} + \varepsilon_{it}|j$
- Exponential regression, $y_{it}|j = \exp(\boldsymbol{\beta}_j'\mathbf{x}_{it}) + \varepsilon_{it}|j$
- Binary choice: probit, logit, complementary log log, Gompertz
- All ordered probability models, probit, logit, etc.
- Count data: Poisson and negative binomial
- Zero inflated count models, Poisson ZIP, negative binomial ZINB
- Censored dependent variable: tobit, grouped data
- Truncated dependent variable: truncated regression
- Loglinear dependent variable: exponential, gamma, inverse Gaussian regression, Weibull, binomial, power, geometric, beta
- Stochastic frontier models: half normal, exponential, Battese/Coelli
- Parametric survival models: Weibull, exponential, lognormal, loglogistic, inverse Gauss

Note that the zero inflation models, Poisson (ZIP), negative binomial (ZINB) and ordered probability models (ZIOP) are latent class models in their own right, outside the estimator being considered here.

R25.4 Output and Saved Results

Estimation results for this estimation program will contain initial results for the model without the latent class treatment, followed by the full set of results for the latent class model. Estimates retained by this model include

Matrices:	*b*	= full parameter vector, $[\boldsymbol{\beta}_1', \boldsymbol{\beta}_2',\ldots F_1,\ldots,F_J]$
	varb	= full covariance matrix

Note that *b* and *varb* involve $J\times(K+1)$ estimates. Two additional matrices are created,

b_class	= a $J\times K$ matrix with each row equal to the corresponding $\boldsymbol{\beta}_j$
class_pr	= a $J\times 1$ vector containing the estimated class probabilities

A third matrix,

beta_i	= an $N\times K$ matrix with one row of estimates for each person

is saved if you include **; Parameters** in the command. See below for discussion

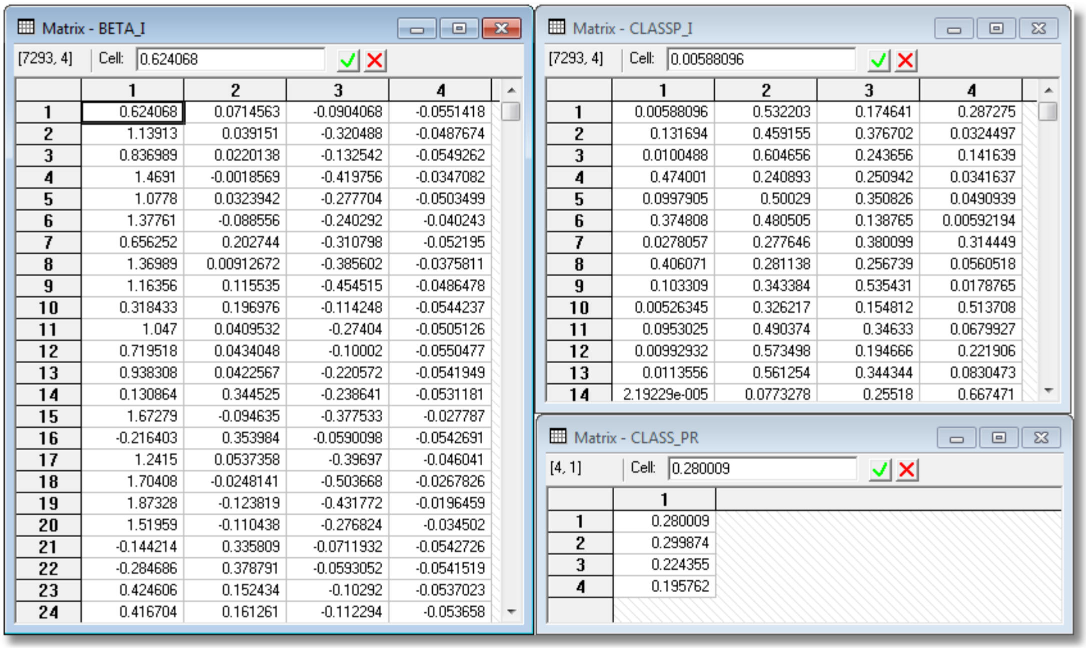
Scalars:	*kreg*	= number of variables in Rhs list
	nreg	= total number of observations used for estimation
	logl	= maximized value of the log likelihood function
	exitcode	= exit status of the estimation procedure.

An example of a four class model based on the German health care data analyzed in Chapter E2 and elsewhere appears below.

```
SAMPLE      ; All $
SETPANEL    ; Group = id ; Pds = ti $
PROBIT      ; Lhs = doctor
            ; Rhs = one,hhninc,hhkids,educ
            ; LCM
            ; Pts = 4
            ; Maxit = 150
            ; Parameters
            ; Panel $
```

```
+-----------------------------------------------------------------+
| Variable = ____________ Variable Groups    Max     Min   Average |
| TI          Group sizes  ID          7293      7       1       3.7 |
+-----------------------------------------------------------------+

-----------------------------------------------------------------------------
Probit   Regression Start Values for DOCTOR
Dependent variable               DOCTOR
Log likelihood function    -17835.48615
Estimation based on N =  27326, K =   4
Inf.Cr.AIC  =35678.972 AIC/N =    1.306
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|     .89068***      .03921    22.72  .0000      .81383    .96754
  HHNINC|    -.04608         .04536    -1.02  .3097     -.13497    .04282
  HHKIDS|    -.22638***      .01576   -14.36  .0000     -.25727   -.19549
    EDUC|    -.03978***      .00342   -11.62  .0000     -.04650   -.03307
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

-----------------------------------------------------------------------------
Latent Class / Panel Probit   Model
Dependent variable               DOCTOR
Log likelihood function    -16383.65581
Restricted log likelihood  -17835.48615
Chi squared [  16 d.f.]      2903.66067
Significance level               .00000
McFadden Pseudo R-squared      .0814012
Estimation based on N =  27326, K =  19
Inf.Cr.AIC  =32805.312 AIC/N =    1.201
Unbalanced panel has   7293 individuals
PROBIT (normal)  probability model
Model fit with  4 latent classes.
-----------------------------------------------------------------------------
```

```
---------+--------------------------------------------------------------
         |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient        Error        z    |z|>Z*        Interval
---------+--------------------------------------------------------------
         |Model parameters for latent class 1
Constant|     2.01229***      .23864     8.43  .0000     1.54457   2.48001
  HHNINC|     -.11883         .27690     -.43  .6678     -.66154    .42388
  HHKIDS|     -.50878***      .14213    -3.58  .0003     -.78735   -.23020
    EDUC|     -.01378         .02016     -.68  .4944     -.05330    .02574
         |Model parameters for latent class 2
Constant|      .96086***      .21747     4.42  .0000      .53463   1.38709
  HHNINC|     -.19837         .29576     -.67  .5024     -.77805    .38131
  HHKIDS|      .13957         .29302      .48  .6339     -.43474    .71387
    EDUC|     -.05816***      .01569    -3.71  .0002     -.08892   -.02740
         |Model parameters for latent class 3
Constant|     1.18089***      .30725     3.84  .0001      .57869   1.78309
  HHNINC|      .35242         .36659      .96  .3364     -.36609   1.07093
  HHKIDS|     -.83844*        .44384    -1.89  .0589    -1.70835    .03146
    EDUC|     -.04909**       .01942    -2.53  .0115     -.08716   -.01103
         |Model parameters for latent class 4
Constant|     -.36679*        .18732    -1.96  .0502     -.73393    .00035
  HHNINC|      .40442**       .17905     2.26  .0239      .05348    .75535
  HHKIDS|     -.05314         .07418     -.72  .4738     -.19853    .09226
    EDUC|     -.05407***      .01544    -3.50  .0005     -.08434   -.02381
         |Estimated prior probabilities for class membership
Class1Pr|      .28001***      .04017     6.97  .0000      .20127    .35875
Class2Pr|      .29987*        .17320     1.73  .0834     -.03960    .63935
Class3Pr|      .22435         .17602     1.27  .2025     -.12065    .56936
Class4Pr|      .19576***      .02528     7.74  .0000      .14621    .24531
--------------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
--------------------------------------------------------------------------
Elapsed time:     0 hours,  0 minutes, 34.03 seconds.
```



Matrix - BETA_I

[7293, 4] Cell: 0.624068

	1	2	3	4
1	0.624068	0.0714563	-0.0904068	-0.0551418
2	1.13913	0.039151	-0.320488	-0.0487674
3	0.836989	0.0220138	-0.132542	-0.0549262
4	1.4691	-0.0018569	-0.419756	-0.0347082
5	1.0778	0.0323942	-0.277704	-0.0503499
6	1.37761	-0.088556	-0.240292	-0.040243
7	0.656252	0.202744	-0.310798	-0.052195
8	1.36989	0.00912672	-0.385602	-0.0375811
9	1.16356	0.115535	-0.454515	-0.0486478
10	0.318433	0.196976	-0.114248	-0.0544237
11	1.047	0.0409532	-0.27404	-0.0505126
12	0.719518	0.0434048	-0.10002	-0.0550477
13	0.938308	0.0422567	-0.220572	-0.0541949
14	0.130864	0.344525	-0.238641	-0.0531181
15	1.67279	-0.094635	-0.377533	-0.027787
16	-0.216403	0.353984	-0.0590098	-0.0542691
17	1.2415	0.0537358	-0.39697	-0.046041
18	1.70408	-0.0248141	-0.503668	-0.0267826
19	1.87328	-0.123819	-0.431772	-0.0196459
20	1.51959	-0.110438	-0.276824	-0.034502
21	-0.144214	0.335809	-0.0711932	-0.0542726
22	-0.284686	0.378791	-0.0593052	-0.0541519
23	0.424606	0.152434	-0.10292	-0.0537023
24	0.416704	0.161261	-0.112294	-0.053658

Matrix - CLASSP_I

[7293, 4] Cell: 0.00588096

	1	2	3	4
1	0.00588096	0.532203	0.174641	0.287275
2	0.131694	0.459155	0.376702	0.0324497
3	0.0100488	0.604656	0.243656	0.141639
4	0.474001	0.240893	0.250942	0.0341637
5	0.0997905	0.50029	0.350826	0.0490939
6	0.374808	0.480505	0.138765	0.00592194
7	0.0278057	0.277646	0.380099	0.314449
8	0.406071	0.281138	0.256739	0.0560518
9	0.103309	0.343384	0.535431	0.0178765
10	0.00526345	0.326217	0.154812	0.513708
11	0.0953025	0.490374	0.34633	0.0679927
12	0.00992932	0.573498	0.194666	0.221906
13	0.0113556	0.561254	0.344344	0.0830473
14	2.19229e-005	0.0773278	0.25518	0.667471

Matrix - CLASS_PR

[4, 1] Cell: 0.280009

	1
1	0.280009
2	0.299874
3	0.224355
4	0.195762

Figure R25.1 Matrix Results for Latent Class Models

R25.5 Extending the Class Probability Model

The latent class probabilities are parameterized in the form of a logit model,

$$F_j = \frac{\exp(\theta_j)}{\Sigma_{m=1}^{J} \exp(\theta_m)},\ \theta_J = 0$$

The latent heterogeneity model can be extended by allowing measured influences in the prior probability. Let z_{i1}, ..., z_{im} denote M time invariant variables (such as sex, marital status, location, education) which affect the latent class probabilities. Then, we extend the model so that prior class assignment is formulated as a multinomial logit;

$$P[\text{class } j \mid \mathbf{z}_i] = F_{ij} = \frac{\exp(\boldsymbol{\theta}_j'\mathbf{z}_i)}{\sum_{m=1}^{J} \exp(\boldsymbol{\theta}_m'\mathbf{z}_i)},\ \boldsymbol{\theta}_J = \mathbf{0}$$

To use this form of the model, change the model command to include the variables

; LCM = the list of variables

Do not include *one* among the variables.

To illustrate, we have extended the example above to allow the class probabilities to vary with age and sex. The new command is

```
PROBIT   ; Lhs = doctor
         ; Rhs = one,hhninc,hhkids,educ
         ; LCM = female,age
         ; Pts = 4
         ; Pds = _groupti $
```

The results now contain two sets of results, the probit equation for each latent class and the multinomial logit model for the class probabilities. There is a well defined likelihood ratio test for the demographic effects in the context of a particular latent class model. The model of the preceding section is nested within this one, so one can use an LR test for the effects. The test statistic in our example would be 2×(-16117.8 – (-16383.66)) = 531.72. There would be eight degrees of freedom. This chi squared is far larger than the tabled critical value, so the hypothesis of the model with constant class probabilities would be rejected.

```
-----------------------------------------------------------------------------
Latent Class / Panel Probit   Model
Dependent variable               DOCTOR
Log likelihood function    -16117.88464
Restricted log likelihood  -17835.48615
Chi squared [  24 d.f.]      3435.20301
Significance level               .00000
McFadden Pseudo R-squared      .0963025
Estimation based on N =  27326, K =  25
Inf.Cr.AIC  =32285.769 AIC/N =    1.182
Unbalanced panel has   7293 individuals
PROBIT (normal)  probability model
Model fit with  4 latent classes.
---------------------------------------------------------------------
```

```
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
  DOCTOR|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Model parameters for latent class 1
Constant|    1.07285***      .32627     3.29  .0010      .43337   1.71233
  HHNINC|    -.39763         .34938    -1.14  .2551    -1.08241    .28714
  HHKIDS|    -.41370***      .11466    -3.61  .0003     -.63844   -.18896
    EDUC|     .06894**       .03306     2.09  .0370      .00415    .13373
        |Model parameters for latent class 2
Constant|     .61316***      .11051     5.55  .0000      .39655    .82976
  HHNINC|     .14997         .10850     1.38  .1669     -.06269    .36263
  HHKIDS|    -.17793***      .03766    -4.73  .0000     -.25173   -.10412
    EDUC|    -.03110***      .00772    -4.03  .0001     -.04624   -.01597
        |Model parameters for latent class 3
Constant|    1.39136***      .35630     3.91  .0001      .69303   2.08969
  HHNINC|     .23213         .38590      .60  .5475     -.52422    .98848
  HHKIDS|     .01433         .13113      .11  .9130     -.24267    .27134
    EDUC|    -.02403         .02580     -.93  .3517     -.07459    .02654
        |Model parameters for latent class 4
Constant|    -.51211***      .16401    -3.12  .0018     -.83356   -.19066
  HHNINC|     .38605**       .16197     2.38  .0172      .06860    .70351
  HHKIDS|    -.04868         .06313     -.77  .4406     -.17241    .07504
    EDUC|    -.04087***      .01347    -3.03  .0024     -.06726   -.01448
        |Estimated prior probabilities for class membership
   ONE_1|   -4.51574***      .59041    -7.65  .0000    -5.67291  -3.35856
FEMALE_1|     .90950***      .18200     5.00  .0000      .55278   1.26623
   AGE_1|     .09563***      .01056     9.06  .0000      .07493    .11632
   ONE_2|     .67420**       .33444     2.02  .0438      .01871   1.32968
FEMALE_2|     .58395***      .15575     3.75  .0002      .27869    .88922
   AGE_2|    -.00244         .00743     -.33  .7429     -.01701    .01213
   ONE_3|     .81172         .88536      .92  .3592     -.92355   2.54698
FEMALE_3|    2.66204***      .35274     7.55  .0000     1.97068   3.35340
   AGE_3|    -.07800***      .03013    -2.59  .0096     -.13706   -.01895
   ONE_4|        0.0    .....(Fixed Parameter).....
FEMALE_4|        0.0    .....(Fixed Parameter).....
   AGE_4|        0.0    .....(Fixed Parameter).....
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
Fixed parameter ... is constrained to equal the value or
had a nonpositive st.error because of an earlier problem.
-----------------------------------------------------------------------------


+-------------------------------------------------------------+
|  Prior class probabilities at data means for LCM variables |
|   Class 1     Class 2     Class 3     Class 4     Class 5  |
|    .23144      .49772      .05760      .21324      .00000  |
+-------------------------------------------------------------+

Elapsed time:     0 hours,  0 minutes, 43.05 seconds.
```

R25.6 Testing for the Latent Class Model

In order to test for latent class effects, you must compare a model with the effects to one without. This is not a straightforward parametric restriction on the latent class model. Note, thus, if $\boldsymbol{\theta}_j$ is set equal to zero, this just produces $F_j = 1/J$. Alternatively, forcing all coefficient vectors to equal zero destroys the identifiability of the latent class probabilities – their standard errors will go to $+\infty$. (Try it.) Therefore, in order to test for class effects, the restricted and unrestricted models must be fit separately. One can use a likelihood ratio test, based on the following computations: For the latent class model the unrestricted log likelihood is,

$$\log L_U = \sum_{i=1}^{N} \log \sum_{j=1}^{J} F_{ij} \prod_{t=1}^{T_i} P(i,t|j).$$

For example, for the Poisson or negative binomial model with no latent class sorting, the log likelihood is

$$\log L_R = \sum_{i=1}^{N} \log \prod_{t=1}^{T_i} P(i,t).$$

In both models, observations within the groups are assumed to be independent. Taking logs in the second expression produces the conventional log likelihood function for the count model,

$$\log L_R = \sum_{i=1}^{N} \sum_{t=1}^{T_i} \log P(i,t).$$

Therefore, it appears that a conventional likelihood ratio statistic can be computed. The degrees of freedom would be $(J\text{-}1)(K_z+K)$. The first $(J\text{-}1)K_z$ would be for the free latent class probabilities while the latter $K(J\text{-}1)$ would be for the additional slope parameters in the last J-1 latent classes. The problem with this approach is that the model is not identified under the restrictions, so this is not a conventional LR test. That is, without the latent class sorting, the extra slope parameters cannot be estimated, and without variation across classes in the slope parameters, the class parameters cannot be estimated. The upshot is that if this is a valid LR statistic, then surely the degrees of freedom is fewer than $(J\text{-}1)(K_z+K)$. But, whether it is appears not to be conclusively determined in the literature. (See Heckman and Singer for discussion.)

A related problem concerns finding the right number of classes. A simple likelihood ratio test of the J class model against the J-1 class model is inappropriate because the degrees of freedom for the test is ambiguous. If two classes have identical parameters, then the model has one less class regardless of whether the two class probabilities are equal or not. So, the degrees of freedom for the test is unclear. In another direction, one cannot test ‘up’ to the number of classes, say starting with J, then incrementing to J+1 if the likelihood increases sufficiently. The reason is that under the alternative (J+1), the lower level estimates are inconsistent. For better or worse, in recent research, analysts have often done this specification search by testing ‘down’ from a model believed to be too large to a smaller one, using not the likelihood ratio statistic, but one of the information criteria, such as the Akaike information measure.

R25.7 Individual Specific Estimates

Among the useful results of this formulation is a posterior estimate of the probabilities of particular group membership; using Bayes theorem,

$$P(j \mid i) \;=\; P(i,j) \,/\, P(i) \;=\; \frac{P(i \mid j)F_{ij}}{\sum_{j=1}^{J} P(i \mid j)F_{ij}}$$

Using this result, we can then compute *j** = the index of the group with the highest posterior probability. Predicted values, residuals, and predicted probabilities for the observed outcomes are then computed as those associated with group *j**. That is, for example,

Fitted value$_{it}$ = conditional mean function|*j**, which will be model specific,

and so on. The **; List** request in a model command will produce a listing of these results, such as that shown in the small example below for a three class model:

```
================================================================================
Predictions computed for the group with the largest posterior probability
================================================================================
Ind.=    1  Most likely group=2  P(j)= .44531  .55469  .00000
Ind.=    2  Most likely group=1  P(j)= .61782  .38218  .00000
Ind.=    3  Most likely group=1  P(j)= .98831  .01169  .00000
Ind.=    4  Most likely group=1  P(j)= .99278  .00722  .00000
Ind.=    5  Most likely group=3  P(j)= .00084  .01151  .98765
Ind.=    6  Most likely group=1  P(j)= .98695  .01305  .00000
Ind.=    7  Most likely group=1  P(j)= .68889  .31111  .00000
Ind.=    8  Most likely group=1  P(j)= .74947  .25053  .00000
Ind.=    9  Most likely group=1  P(j)= .99247  .00753  .00000
Ind.=   10  Most likely group=1  P(j)= .95551  .04449  .00000
(Rows 11-25 omitted)
Ind.=   26  Most likely group=1  P(j)= .82398  .17602  .00000
Ind.=   27  Most likely group=1  P(j)= .96988  .03012  .00000
Ind.=   28  Most likely group=2  P(j)= .02913  .97087  .00000
Ind.=   29  Most likely group=2  P(j)= .16479  .83521  .00000
Ind.=   30  Most likely group=1  P(j)= .84708  .15292  .00000
```

The preceding also suggests a person specific estimate of the parameter vector. The prior estimate would be the one from the most likely class, based only on the estimates parameters. But, using all the information available for the individual, we can compute a conditional mean of the posterior distribution using

$$\hat{\boldsymbol{\beta}}_i = E[\boldsymbol{\beta}_i \mid data_i, \hat{\boldsymbol{\theta}}] = \sum_{j=1}^{J} \hat{P}(j \mid i)\hat{\boldsymbol{\beta}}_j$$

We emphasize, this is an estimator of the mean of a conditional distribution in exactly the same fashion as discussed in Section R24.5. It is not truly an estimator of a person specific parameter vector. For better or worse, perhaps the best estimator of that would be $\hat{\boldsymbol{\beta}}_{j*}$, the estimated parameter vector associated with the most likely class.

Use the **; Parameters** specification in your model command to request this computation. This will save a matrix named *beta_i* containing the estimates. Note, this matrix may be quite large, as there is one vector for each individual in the sample – each person is a row in this matrix. An example appears above in Section R25.4. See Figure R25.1.

R25.7.1 Individual Specific Posterior Class Probabilities

The posterior probabilities for the latent classes may be saved as variables in the data set rather than in a matrix. To do so, you must have variables in the data set which will be given the probabilities. The following general form will create columns in your data set containing missing values, ready to receive the probabilities. There must then be a namelist to collect the names, such as the one below,

CREATE ; name1,name2,…,nameJ $
NAMELIST ; probs = name1,...,nameJ $ (Use any name you wish.)

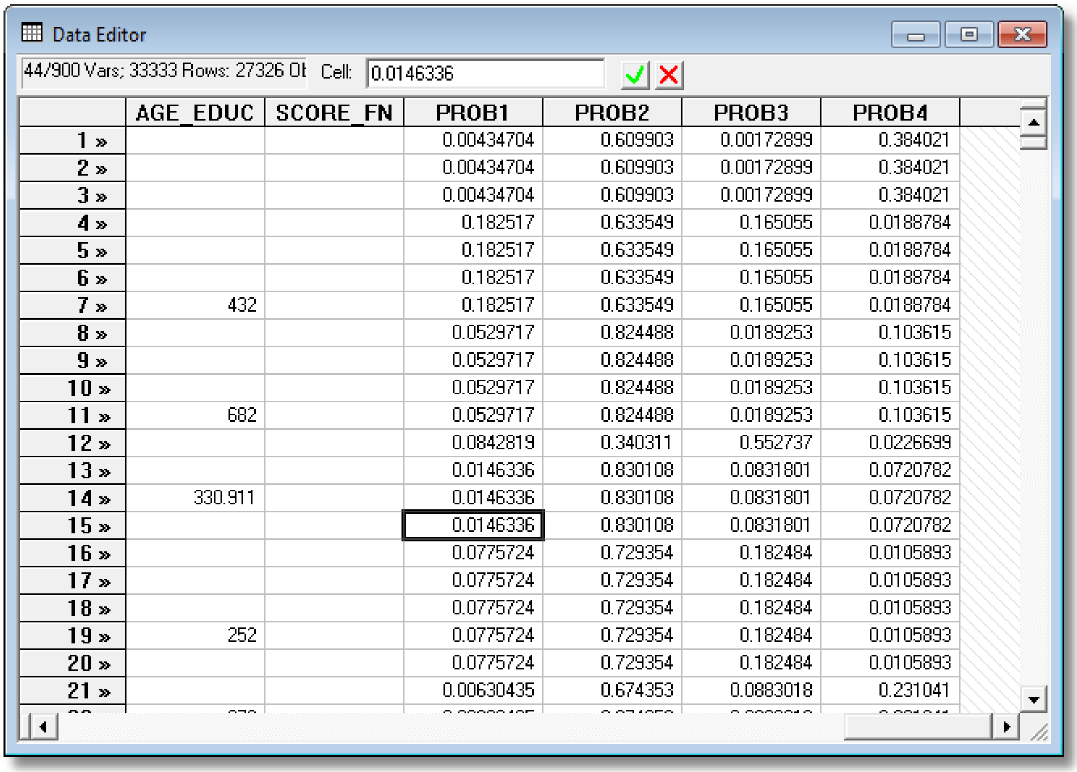
Finally, add

; Classp = the name of the namelist

to the model command. We changed the example in Section R25.4 by adding

CREATE ; prob1,prob2,prob3,prob4 $
NAMELIST ; probs = prob1,prob2,prob3,prob4 $

then added **; Classp = probs** to the model command. The new data, shown in Figure R25.2, results.



44/900 Vars; 33333 Rows: 27326 Ol Cell: 0.0146336

	AGE_EDUC	SCORE_FN	PROB1	PROB2	PROB3	PROB4
1 »			0.00434704	0.609903	0.00172899	0.384021
2 »			0.00434704	0.609903	0.00172899	0.384021
3 »			0.00434704	0.609903	0.00172899	0.384021
4 »			0.182517	0.633549	0.165055	0.0188784
5 »			0.182517	0.633549	0.165055	0.0188784
6 »			0.182517	0.633549	0.165055	0.0188784
7 »	432		0.182517	0.633549	0.165055	0.0188784
8 »			0.0529717	0.824488	0.0189253	0.103615
9 »			0.0529717	0.824488	0.0189253	0.103615
10 »			0.0529717	0.824488	0.0189253	0.103615
11 »	682		0.0529717	0.824488	0.0189253	0.103615
12 »			0.0842819	0.340311	0.552737	0.0226699
13 »			0.0146336	0.830108	0.0831801	0.0720782
14 »	330.911		0.0146336	0.830108	0.0831801	0.0720782
15 »			0.0146336	0.830108	0.0831801	0.0720782
16 »			0.0775724	0.729354	0.182484	0.0105893
17 »			0.0775724	0.729354	0.182484	0.0105893
18 »			0.0775724	0.729354	0.182484	0.0105893
19 »	252		0.0775724	0.729354	0.182484	0.0105893
20 »			0.0775724	0.729354	0.182484	0.0105893
21 »			0.00630435	0.674353	0.0883018	0.231041

Figure R25.2 Individual Specific Posterior Probabilities

R25.7.2 Individual Specific Parameters

The matrix *beta_i* that is saved by **; Parameters** contains a full set of estimates for each individual or group in the sample. If you have a large number of individuals or a large model or both, this may quickly exhaust the 50,000 cell limit on a saved matrix. You can use the same procedure detailed in the previous section to save specific parameters in the data area instead of in a matrix. The procedure is once again to create the template variables and add them to a namelist, then add

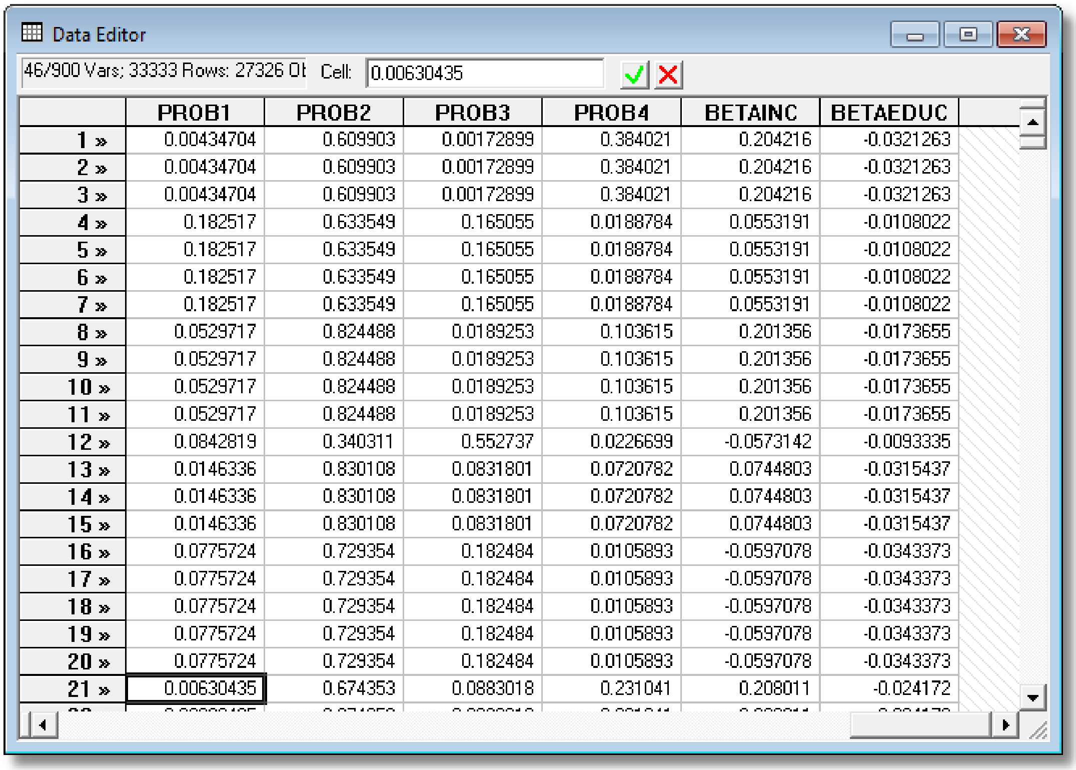
; Par = namelist(lclist)

To the model command, where the namelist is the one just created and the *lclist* is the names of the specific variables in the model. For an example, we used

```
CREATE      ; betainc, betaeduc $
NAMELIST    ; betalcm = betainc,betaeduc $
```

Then **; Par = betalcm (hhninc,educ)**

The results are shown in Figure R25.3.



Data Editor

46/900 Vars; 33333 Rows: 27326 Ot   Cell: 0.00630435

	PROB1	PROB2	PROB3	PROB4	BETAINC	BETAEDUC
1 »	0.00434704	0.609903	0.00172899	0.384021	0.204216	-0.0321263
2 »	0.00434704	0.609903	0.00172899	0.384021	0.204216	-0.0321263
3 »	0.00434704	0.609903	0.00172899	0.384021	0.204216	-0.0321263
4 »	0.182517	0.633549	0.165055	0.0188784	0.0553191	-0.0108022
5 »	0.182517	0.633549	0.165055	0.0188784	0.0553191	-0.0108022
6 »	0.182517	0.633549	0.165055	0.0188784	0.0553191	-0.0108022
7 »	0.182517	0.633549	0.165055	0.0188784	0.0553191	-0.0108022
8 »	0.0529717	0.824488	0.0189253	0.103615	0.201356	-0.0173655
9 »	0.0529717	0.824488	0.0189253	0.103615	0.201356	-0.0173655
10 »	0.0529717	0.824488	0.0189253	0.103615	0.201356	-0.0173655
11 »	0.0529717	0.824488	0.0189253	0.103615	0.201356	-0.0173655
12 »	0.0842819	0.340311	0.552737	0.0226699	-0.0573142	-0.0093335
13 »	0.0146336	0.830108	0.0831801	0.0720782	0.0744803	-0.0315437
14 »	0.0146336	0.830108	0.0831801	0.0720782	0.0744803	-0.0315437
15 »	0.0146336	0.830108	0.0831801	0.0720782	0.0744803	-0.0315437
16 »	0.0775724	0.729354	0.182484	0.0105893	-0.0597078	-0.0343373
17 »	0.0775724	0.729354	0.182484	0.0105893	-0.0597078	-0.0343373
18 »	0.0775724	0.729354	0.182484	0.0105893	-0.0597078	-0.0343373
19 »	0.0775724	0.729354	0.182484	0.0105893	-0.0597078	-0.0343373
20 »	0.0775724	0.729354	0.182484	0.0105893	-0.0597078	-0.0343373
21 »	0.00630435	0.674353	0.0883018	0.231041	0.208011	-0.024172

Figure R25.3 Individual Specific Parameters

R25.8 Application

The preceding displays a fairly detailed example of an estimated latent class model. To illustrate the technique further, we will show an approach for handling the canonical, original problem of latent class analysis, separating a mixture of normal distributions. We consider a sample, $y_1,...,y_N$ in which the data generating mechanism for the data is a pair of latent normal distributions with equal variances but different means. The following procedure can be used to carry out experiments for this problem. The data generation mechanism creates a mixture of normals, N(0,1) with probability prob and N(μ,1) with probability 1-prob. We display a kernel density estimator of the mixed distribution, then use a linear regression on a constant to estimate the class probabilities and underlying means. It can be seen by trying different values that the mixture estimator is more successful the more sharply defined are the underlying data (of course). In the first experiment below, the estimator is unable to distinguish the two classes. The second works out much more favorably.

```
PROC = LCM(mu,prob) $
SAMPLE      ; 1-1000 $
CALC        ; Ran(123457) $
CREATE      ; mix = Rnu(0,1) $
CREATE      ; If(mix < prob)  y = Rnn(0,1)
            ; (Else)          y = Rnn(mu,1) $
KERNEL      ; Rhs = y $
REGRESS     ; Lhs = y ; Rhs = one
            ; LCM
            ; Pts = 2 $
ENDPROC $
EXEC        ; Proc = LCM(1.5, .5) $
EXEC        ; Proc = LCM(3,.4) $
```

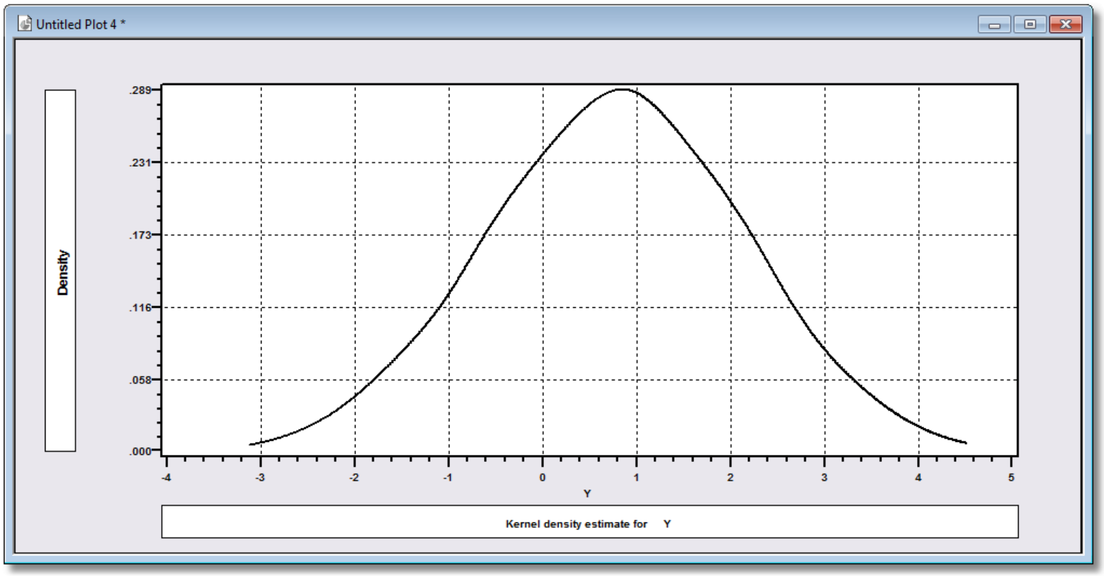


Figure R25.4 Kernel Density Estimator for y

```
+--------------------------------------+
| Kernel Density Estimator for Y       |
| Observations       =         1000    |
| Points plotted     =         1000    |
| Bandwidth          =      .410767    |
| Statistics for abscissa values----   |
| Mean               =     1.839709    |
| Standard Deviation =     1.816993    |
| Minimum            =    -2.836608    |
| Maximum            =     5.727679    |
| ---------------------------------    |
| Kernel Function    =     Logistic    |
| Cross val. M.S.E.  =      .000000    |
| Results matrix     =       KERNEL    |
+--------------------------------------+
Elapsed time:     0 hours,  0 minutes,   .13 seconds.

-----------------------------------------------------------------------------
OLS Starting values for latent classes model......
Ordinary     least squares regression ............
LHS=Y        Mean                 =       1.83971
             Standard deviation   =       1.81699
             Number of observs.   =          1000
Model size   Parameters           =             1
             Degrees of freedom   =           999
Residuals    Sum of squares       =       3298.16
             Standard error of e  =       1.81699
Fit          R-squared            =        .00000
             Adjusted R-squared   =        .00000
Diagnostic   Log likelihood       =   -2015.62129
             Restricted(b=0)      =   -2015.62129
Info criter. Akaike Info. Criter. =       1.19537
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
Constant|    1.83971***      .05746    32.02  .0000     1.72709   1.95233
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

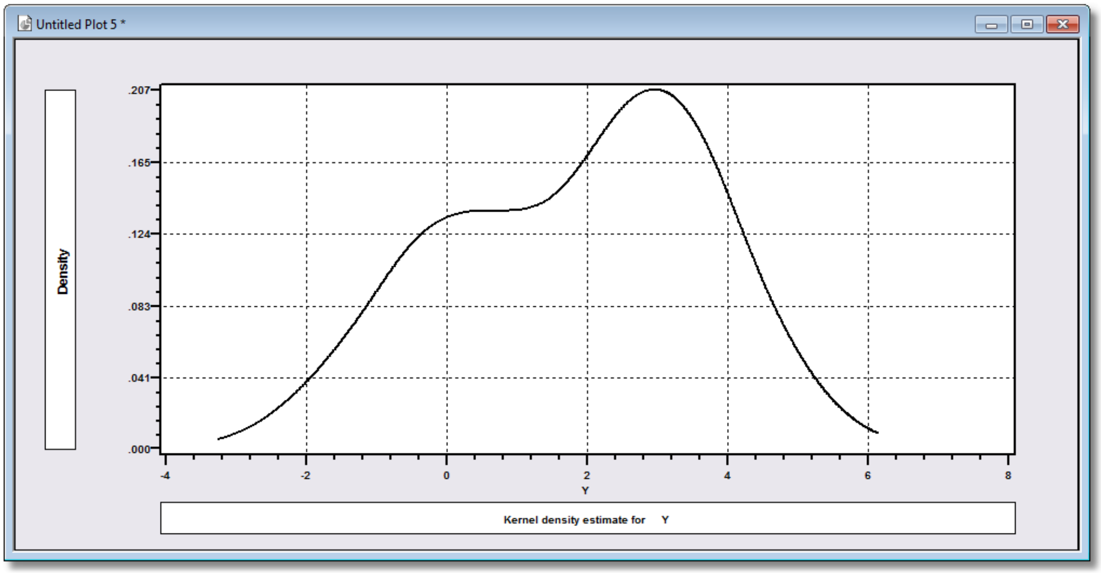


Figure R25.5 Kernel Density Estimator for y

```
Normal exit:  20 iterations. Status=0, F=    1943.114

-----------------------------------------------------------------------------
Latent Class / Panel LinearRg Model
Dependent variable                     Y
Log likelihood function     -1943.11398
Restricted log likelihood  -17835.48615
Chi squared [   4 d.f.]     31784.74433
Significance level               .00000
McFadden Pseudo R-squared      .8910535
Estimation based on N =   1000, K =   5
Inf.Cr.AIC  = 3896.228 AIC/N =    3.896
Sample is  1 pds and   1000 individuals
LINEAR regression model
Model fit with  2 latent classes.
--------+--------------------------------------------------------------------
        |                  Standard            Prob.      95% Confidence
       Y|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Model parameters for latent class 1
Constant|    3.08638***      .07717    40.00  .0000     2.93513   3.23763
   Sigma|     .96456***      .05246    18.39  .0000      .86175   1.06738
        |Model parameters for latent class 2
Constant|     .01659         .12197      .14  .8918     -.22246    .25565
   Sigma|    1.07898***      .08190    13.17  .0000      .91846   1.23950
        |Estimated prior probabilities for class membership
Class1Pr|     .59389***      .03127    18.99  .0000      .53260    .65518
Class2Pr|     .40611***      .03127    12.99  .0000      .34482    .46740
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------

Elapsed time:     0 hours,  0 minutes,   .22 seconds.

Maximum repetitions of PROC
```

R25.9 Technical Details on Estimating Latent Class Models

The sequence of T_i observations for individual i, given group j is $\mathbf{y}(i|j)$ = $[y(i,1|j),y(i,2|j),...,y(i,T_i|j)]$. Observations for individual i in different periods are assumed to be independent. Thus, the joint probability of the sequence of observations $[\mathbf{y}(i|j)]$ is

$$P(i|j) \quad = \quad \prod_{t=1}^{T_i} \; P(i,t|j).$$

We denote the mass, or probability in interval (group) j as F_{ij}, $j = 1,...,J$, such that $F_{i1} + F_{i2} + ... + F_{iJ}$ = 1. Then, the posterior probability of an observed sequence of observations is

$$P(i) \quad = \quad \sum_{j=1}^{J} \; F_j P(i|j)$$

where F_j is the probability of membership in the jth class. We parameterize the group probabilities with

$$F_{ij} \quad = \quad \frac{\exp(\boldsymbol{\theta}_j' \mathbf{z}_i)}{\sum_{j=1}^{J} \exp(\boldsymbol{\theta}_j' \mathbf{z}_i)}$$

where $\boldsymbol{\theta}_J = \mathbf{0}$, since $\Sigma_j F_{ij} = 1$. The log likelihood function for the observed sample is

$$\begin{aligned}\log L \quad &= \sum_{i=1}^{N} \; \log[P(i)] \\ &= \sum_{i=1}^{N} \; \log \sum_{j=1}^{J} \; F_j \prod_{t=1}^{T_i} \; P(i,t|j)\end{aligned}$$

This function is maximized with respect to the vector of parameters $\boldsymbol{\beta} = (\boldsymbol{\beta}_1,...,\boldsymbol{\beta}_J)$, $\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_J$. subject to the restriction that $\boldsymbol{\theta}_J = 0$. (Other restrictions may be imposed as well.)

Maximization of the log likelihood does not require any unusual techniques or approaches. (Some authors, e.g., Cockburn (1999) have used the EM algorithm for a Poisson model of this sort, but this is a means to an end, not a necessity. We have found that the conventional approach used here works without problems, and is much simpler.) The gradient of the log likelihood function is

$$\begin{aligned}\frac{\partial \log L}{\partial \boldsymbol{\beta}_j} &= \sum_{i=1}^{N} \frac{1}{P_i} F_{ij} P_{i|j} \sum_{t=1}^{T_i} \frac{\partial \log P_{it|j}}{\partial \boldsymbol{\beta}_j} &&= \sum_{i=1}^{N} \frac{F_{ij} P_{i|j}}{P_i} \sum_{t=1}^{T_i} \mathbf{g}_{it|j} \\ \frac{\partial \log L}{\partial \boldsymbol{\theta}_j} &= \sum_{i=1}^{N} \frac{1}{P_i} \sum_{m=1}^{J} P_{i|m} F_{im} [1(j=m) - F_{im}]\,\mathbf{z}_i &&= \sum_{i=1}^{N} \sum_{m=1}^{J} \frac{F_{im} P_{i|j}}{P_i} \mathbf{r}_{im}\end{aligned}$$

The gradients in the first term are the ordinary derivatives of the log probabilities that enter the log likelihood in the other formulations we have considered. The asymptotic covariance matrix for the estimated parameters is computed from the estimated Hessian of the log likelihood. The Hessian is obtained as follows, where we will sketch the derivation and skip a bit of the algebra. The mixed derivative, $\partial^2 \log L/\partial \boldsymbol{\beta}_j \partial \boldsymbol{\beta}_k'$, is simplified by the fact that $P_{it|j}$ involves only a single parameter vector, so the only second derivatives with respect to model parameters involve products of first derivatives. Also, F_{ij} is not a function of $\boldsymbol{\beta}_j$. Let $\mathbf{H}_{it|j}$ be the 'own' Hessian,

$$\mathbf{H}_{it|j} = \frac{\partial^2 \log L}{\partial \boldsymbol{\beta}_j \partial \boldsymbol{\beta}_j'}.$$

The gradient above may be written

$$\frac{\partial \log L}{\partial \boldsymbol{\delta}} = \sum\nolimits_{i=1}^{n} \mathbf{g}_i$$

where $\boldsymbol{\delta}$ is the full set of parameters in the model. The BHHH estimator for estimating the asymptotic covariance matrix would be based on

$$\mathbf{D}^{-1} = \left(\sum\nolimits_{i=1}^{N} \mathbf{g}_i \mathbf{g}_i'\right)^{-1}$$

The results just cited imply that the Hessian of the log likelihood is equal to **D** plus a set of terms we now define. The diagonal block $\partial^2 \log L/\partial \boldsymbol{\beta}_j \partial \boldsymbol{\beta}_i'$ is

$$\mathbf{H}_{\beta j \beta j'} = \mathbf{D}_{\beta\beta'} + \sum\nolimits_{i=1}^{N} \frac{F_{ij} P_{i|j}}{P_i}\left[\sum\nolimits_{t=1}^{T_i} \mathbf{H}_{it|j} + \left(\sum\nolimits_{t=1}^{T_i} \mathbf{g}_{it|j}\right)\left(\sum\nolimits_{t=1}^{T_i} \mathbf{g}_{it|j}\right)'\right]$$

The full Hessian for the parameters in the prior probabilities is augmented, in blocks for θ_j, θ_m by

$$\mathbf{H}_{\theta j\, \theta m'} = \mathbf{D}_{\theta j\, \theta m} + \sum\nolimits_{i=1}^{N} \sum\nolimits_{j=1}^{J} \frac{F_{ij} P_{i|j}}{P_i}\left[\frac{\partial^2 \log F_{ij}}{\partial \boldsymbol{\theta}_j \partial \boldsymbol{\theta}_m'} + \left(\frac{\partial \log F_{ij}}{\partial \boldsymbol{\theta}_j}\right)\left(\frac{\partial \log F_{ij}}{\partial \boldsymbol{\theta}_m'}\right)\right]$$

Finally, the cross derivative for β_j and θ_m is

$$\mathbf{H}_{\beta j \beta j'} = \mathbf{D}_{\theta j\, \theta m} + \sum\nolimits_{i=1}^{N} \frac{F_{ij} P_{i|j}}{P_i}\left[\left(\sum\nolimits_{t=1}^{T_i} \mathbf{g}_{it|j}\right)\left(\frac{\partial \log F_{ij}}{\partial \boldsymbol{\theta}_m'}\right)\right].$$

Thus, the Hessian differs from **D**, and the estimator differs from the BHHH estimator, by the augmented terms shown above. Note that each of these should be close to zero, as within the square brackets in the first two cases is the second derivative plus the squared first derivative. This sum should normally have expectation zero if the maximand were the only term in the brackets. Given the structure of the problem, these terms will not be zero. However, this being a regular maximum likelihood problem, the Hessian and the outer products matrix do have the same probability limits. We generally use the inverse of the analytical Hessian for the asymptotic covariance matrix. However, if the model is overspecified (too many classes), this matrix may fail to be positive definite – some roots may become close to zero. In this case, the program automatically reverts to the BHHH estimator, $\mathbf{D}^{-1}$. For starting values for the iterations, we use the single class estimates for the parameters, perturbed slightly (5%) and class probabilities equal to $1/J$.

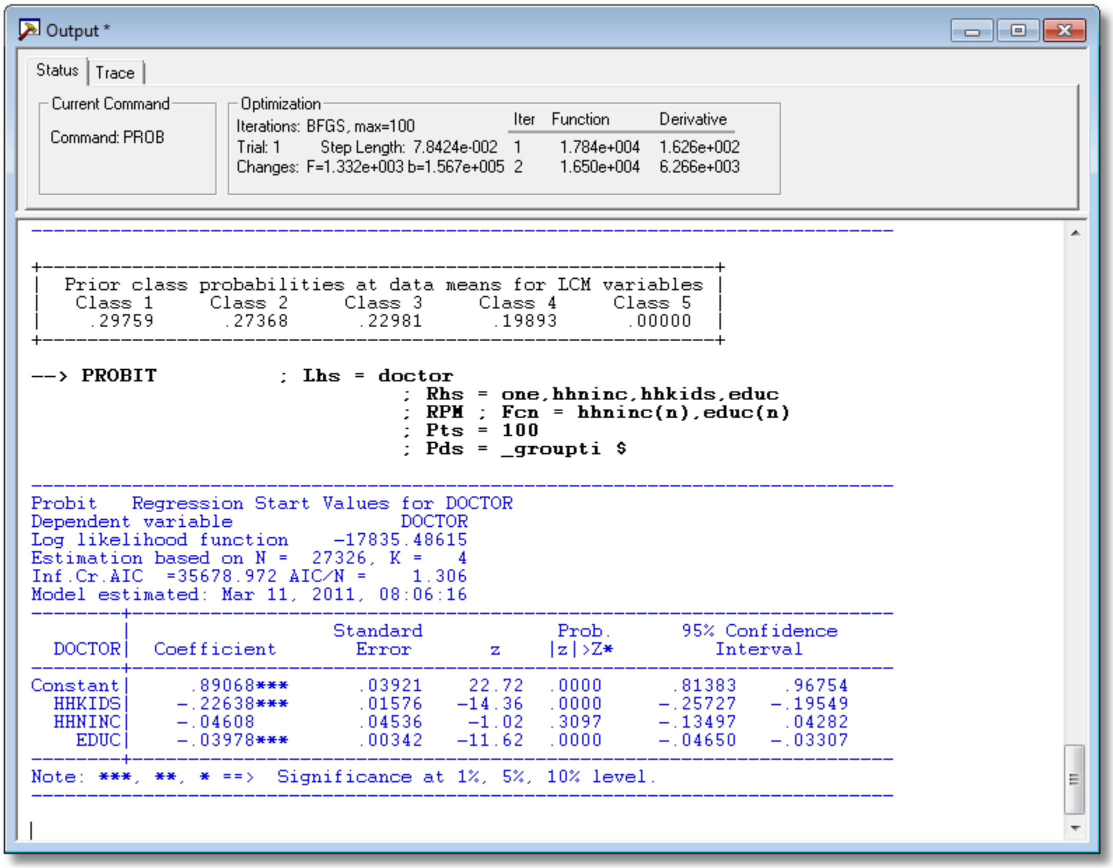
R26: Numerical Optimization

R26.1 Numerical Optimization

Most of *LIMDEP*’s models are quite nonlinear and require iterative procedures for estimating the parameters. In most cases, how this is done need not concern you any more than does the internal method of computing ordinary least squares coefficients. But, there are a few things which will be helpful for you to know about *LIMDEP*’s internal workings. This chapter will describe those aspects of the nonlinear optimization procedures which you can control and give some technical background on the methods that will be useful to you when you need to diagnose why a procedure appears to have failed. Also, Section R26.10 on starting values is especially important, since your starting values will be part of one method of testing hypotheses.

R26.2 Technical Display During Optimization

If your output window has the Status tab selected, then during iterations, you will see on your screen a progress report of the sort shown in Figure R26.1



```
Current Command
Command: PROB

Optimization
Iterations: BFGS, max=100                          Iter  Function    Derivative
Trial: 1       Step Length: 7.8424e-002            1     1.784e+004  1.626e+002
Changes:  F=1.332e+003 b=1.567e+005                2     1.650e+004  6.266e+003
```

```
-----------------------------------------------------------------------------

+-----------------------------------------------------------------+
|  Prior class probabilities at data means for LCM variables |
|    Class 1      Class 2      Class 3      Class 4      Class 5   |
|     .29759       .27368       .22981       .19893       .00000   |
+-----------------------------------------------------------------+

--> PROBIT              ; Lhs = doctor
                                   ; Rhs  = one,hhninc,hhkids,educ
                                   ; RPM  ; Fcn  = hhninc(n),educ(n)
                                   ; Pts  = 100
                                   ; Pds  = _groupti $

-----------------------------------------------------------------------------
Probit   Regression Start Values for DOCTOR
Dependent variable                 DOCTOR
Log likelihood function      -17835.48615
Estimation based on N =   27326, K =    4
Inf.Cr.AIC  =35678.972 AIC/N =     1.306
Model estimated: Mar 11, 2011, 08:06:16
--------+--------------------------------------------------------------------
        |                Standard            Prob.       95% Confidence
  DOCTOR|  Coefficient      Error         z    |z|>Z*        Interval
--------+--------------------------------------------------------------------
Constant|      .89068***     .03921     22.72  .0000      .81383     .96754
  HHKIDS|    -.22638***      .01576    -14.36  .0000    -.25727    -.19549
  HHNINC|    -.04608         .04536     -1.02  .3097    -.13497     .04282
    EDUC|    -.03978***      .00342    -11.62  .0000    -.04650    -.03307
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Figure R26.1 Technical Output During Optimization

The top line reports the model being estimated, the algorithm that is being used and the maximum iterations. There are default algorithms for all procedures, but you can change the algorithm if need be. (This should be rare.) The iteration reports show the function value (usually the negative of the log likelihood function) and the ‘derivative’ which is a rate of change of the gradient. This is one of the convergence criteria discussed below. The first iteration will remain on the screen on the line labeled iteration 1. This shows you the function value at the starting values. Then subsequent iterations are reported on the second line. Thus, this reports the change in the function value from entry to the current iteration. The line marked ‘changes’ reports two other convergence criteria, the rate of change of the function and the maximum proportional change in the parameters. (These are described more fully below.) If you are using the DFP or BFGS algorithm, the next few lines will display the progress of the line search.

R26.3 Technical Output During Iterations

The technical output described in Section R26.4.1 will normally not appear in your output window. If your estimation problem is well specified and appropriate for your data, you generally will not need it. However, it is helpful to have the technical diagnostic information produced by the iterations when the optimization is not going well. For example, you might want to find out why a particular procedure failed to converge. The specification

; Output = n

added to any model command is used to control the amount of intermediate output produced in the output file. The values of *n* and the corresponding output produced at each iteration are shown in the example below. The results were all produced by the same well behaved probit model, with only changes in this one setting: In each case, iterations 3 through 6 are omitted. Note that most of this information is actually displayed in the optimization report shown in Figure R26.1, but only in a rolling ticker format. The following shows how to retain this information for inspection after estimation.

n = 0: No technical information. Only the exit status is given.

```
Normal exit from iterations. Exit status=0.
```

n > 0: Starting values, maximum iterations, convergence rules, and algorithm for all cases

n = 1: Log likelihood only

```
----------------------------------------------------------
Nonlinear Estimation of Model Parameters
Method                                    BFGS
Maximum iterations                         100
Maximum steps in line search                20
Convergence criteria
  Gradient norm                    .10000D-05
  Change in function value         .00000D+00
  Maximum parameter change         .00000D+00
Laguerre                                    20
Hermite                                     64
Replications for GHK simulator=            100
Convergence criteria at iteration   1...
Gradient norm                     .195556D+06
Function change                   .185550D+05
Maximum parameter change          .333183D+08
Convergence criteria at iteration   2...
```

```
Gradient norm                .195281D+05
Function change              .563588D+03
Maximum parameter change     .226661D+08
Convergence criteria at iteration   7...
Gradient norm                .590832D-08[Grdnt Converged]
Function change              .738510D-09
Maximum parameter change     .178605D-07
Iterative procedure has converged
---------------------------------------------------------
Normal exit:   7 iterations. Status=0, F=    .1770360D+05
Function value was   .1855497972D+05 at entry -----------
                     .1770360132D+05 at exit  -----------
```

n = 2 or 3: 1 and first derivative vector

```
---------------------------------------------------------
Nonlinear Estimation of Model Parameters
Method                                BFGS
Maximum iterations                     100
Maximum steps in line search            20
Convergence criteria
  Gradient norm                .10000D-05
  Change in function value     .00000D+00
  Maximum parameter change     .00000D+00
Laguerre                                20
Hermite                                 64
Replications for GHK simulator=        100
=========================================================
Current iteration    1
Parameter          Value      Derivative
B(001)       .513764D+00     .483416D+04
B(002)       .560961D-02     .186903D+06
B(003)      -.113791D-01     .573247D+05
Function value               .185550D+05
Convergence criteria at iteration   1...
Gradient norm                .195556D+06
Function change              .185550D+05
Maximum parameter change     .333183D+08
Current iteration    7
Parameter          Value      Derivative
B(001)       .128477D-01     .401093D-06
B(002)       .152351D-01     .177065D-04
B(003)      -.300083D-01     .305399D-05
Function value               .177036D+05
Convergence criteria at iteration   7...
Gradient norm                .590832D-08[Grdnt Converged]
Function change              .738510D-09
Maximum parameter change     .178605D-07
Iterative procedure has converged
---------------------------------------------------------
Normal exit:   7 iterations. Status=0, F=    .1770360D+05
Function value was   .1855497972D+05 at entry -----------
                     .1770360132D+05 at exit  -----------
```

n = 4: 3 and stepsize search (not used for Newton's method)

```
---------------------------------------------------------
Nonlinear Estimation of Model Parameters
Method                                BFGS
Maximum iterations                     100
Maximum steps in line search          20
Convergence criteria
  Gradient norm                .10000D-05
  Change in function value     .00000D+00
  Maximum parameter change     .00000D+00
Laguerre                              20
Hermite                               64
Replications for GHK simulator=      100
Starting Values------------------------
              B(001)      .513763908D+00
              B(002)      .560960776D-02
              B(003)     -.113790606D-01
=========================================================
Current iteration    1
Parameter          Value      Derivative
B(001)       .513764D+00     .483416D+04
B(002)       .560961D-02     .186903D+06
B(003)      -.113791D-01     .573247D+05
Function value               .185550D+05
Convergence criteria at iteration   1...
Gradient norm                .195556D+06
Function change              .185550D+05
Maximum parameter change     .333183D+08
Line Search------------------------------------------
Try =  0 F=  .1855D+05 Step=  .0000D+00 Slope= -.1956D+06
Try =  1 F=  .1846D+06 Step=  .1000D+00 Slope=  .3515D+07
Try =  2 F=  .1799D+05 Step=  .5264D-02 Slope= -.1679D+05
=========================================================
Current iteration    2
Parameter          Value      Derivative
B(001)       .513634D+00     .104076D+04
B(002)       .578889D-03     .131212D+05
B(003)      -.129220D-01     .144257D+05
Function value               .179914D+05
Convergence criteria at iteration   2...
Gradient norm                .195281D+05
Function change              .563588D+03
Maximum parameter change     .226661D+08
Line Search------------------------------------------
Try =  0 F=  .1799D+05 Step=  .0000D+00 Slope= -.1953D+05
Try =  1 F=  .1823D+05 Step=  .5264D-02 Slope=  .1103D+06
Try =  2 F=  .1798D+05 Step=  .8039D-03 Slope= -.7990D+01
=========================================================
Current iteration    7
Parameter          Value      Derivative
B(001)       .128477D-01     .401093D-06
B(002)       .152351D-01     .177065D-04
B(003)      -.300083D-01     .305399D-05
Function value               .177036D+05
Convergence criteria at iteration   7...
Gradient norm                .590832D-08[Grdnt Converged]
Function change              .738510D-09
Maximum parameter change     .178605D-07
Iterative procedure has converged
---------------------------------------------------------
Normal exit:   7 iterations. Status=0, F=    .1770360D+05
Function value was   .1855497972D+05 at entry -----------
                     .1770360132D+05 at exit  -----------
```

R26.4 Exit from Iterations and Warning Messages

Although the problems *LIMDEP* is programmed to handle are highly nonlinear, they are usually straightforward to solve, and convergence of iterative procedures is usually routine. But, optimization procedures sometimes break down. Unless you have a perfectly collinear data matrix, you can always compute the coefficients of a linear regression model. This is not true of a nonlinear model, and the optimizer can break down for various reasons.

R26.4.1 Normal Exit from Iterations

In theory, one exits the search for a maximizer of a function when the derivatives become zero. But, in digital computing, this never happens – because of rounding error and the way that numbers are represented, the practical rule is that one exits when the *computed* derivatives become small enough, or some other quantity becomes close enough to a theoretical target. Exit criteria – the rules for deciding when proper convergence has been achieved – for leaving iterative procedures are discussed below. Note the convergence assessment in the examples above. Normal exit from iterations is marked clearly in the technical output for the model.

The theoretical solution to the optimization occurs where all three convergence criteria listed are 0.0. The marker after the last iteration shows which criterion indicated the convergence had been satisfactorily achieved to within the acceptable tolerance. The exit code of zero for this procedure is shown in the results. The last line shows the function value – usually the negative of the log likelihood function as it is here – at the starting values, then at the final values. Standard model output consisting of parameter estimates, standard errors, and other statistics will now follow.

R26.4.2 Maximum Iterations

Large numbers of iterations may be a tipoff that something is wrong. *LIMDEP*’s models usually take no more than 25 iterations to fit, and often take far less. Exceptions are latent class and random parameters, certain sample selection models and other models which involve correlation coefficients for two or more normal distributions. These can take 50 or 75. But, users have reported 100, 200, 1,000 and more. It is generally unlikely that any estimator which is going to converge at all would ever reach 100 iterations before doing so, and if you find this is the case with yours, you should check the diagnostic statistics to see if you really have obtained the optimum the estimator was seeking. When the maximum number of iterations is reached before convergence, you will see the following:

```
=========================================================
Current iteration    6
Parameter          Value        Derivative
B(001)       .128477D-01    -.696981D-04
B(002)       .152351D-01    -.290964D-02
B(003)      -.300083D-01    -.404782D-03
Function value                .177036D+05
Maximum of    5 iterations. Exit iterations with status=1
Function value was    .1855497972D+05 at entry -----------
                      .1770360132D+05 at exit  -----------
```

This will be followed by the current set of parameter estimates, with full output as if proper convergence had been reached. You should generally check other results to be sure that this is the case – generally, when the maximum number of iterations is reached, the optimizer has not converged.

R26.4.3 Unable to Find Function Optimum

The number of trials in the line search is another indication of imminent failure. When an estimator which uses a line search method is proceeding smoothly toward its optimum, each step in the line search will take a small number of 'tries,' typically five or less, and usually only two The example above with n = 4 shows an example of a smooth, well behaved line search toward a function optimum.

Occasionally, more than two tries will occur during a line search, even 10 or 15. But, again, this will be unusual. When 20 trials are reached, *LIMDEP* will back up, and try a slightly different direction. But, expect 20 more trials to occur, after which breakdown of the iterations is likely to be next. The next example, which is a deliberately misconstructed optimization problem, illustrates the sort of diagnostic output that will result when the optimization is in the process of failing: (Two iterations which preceded the 'crash' have been omitted.)

```
Nonlinear Estimation of Model Parameters
Method=BFGS  ; Maximum iterations=100
Convergence criteria:gtHg  .1000D-05 chg.F .0000D+00 max|dB|=.0000D+00
Start values:    .10000D+01   .10000D+01   .10000D+01
1st derivs.      .39403D+01   .29861D+03   .97451D+02
Parameters:      .10000D+01   .10000D+01   .10000D+01
Itr  1 F= -.2869D+05 gtHg= .3141D+03 chg.F=.2869D+05 max|db|=.2986D+03
Try =  0 F= -.2869D+05 Step=  .0000D+00 Slope= -.3141D+03
Try =  1 F= -.2872D+05 Step=  .1000D+00 Slope= -.3415D+03
Try =  2 F= -.2876D+05 Step=  .2000D+00 Slope= -.3742D+03
Try =  3 F= -.2884D+05 Step=  .4000D+00 Slope= -.4637D+03
Try =  4 F= -.2683D+05 Step=  .8000D+00 Slope=  .1341D+05
Try =  5 F= -.2886D+05 Step=  .4312D+00 Slope= -.4818D+03
Try =  6 F= -.2887D+05 Step=  .4521D+00 Slope= -.4948D+03
Try =  7 F= -.2888D+05 Step=  .4685D+00 Slope= -.5055D+03
Try =  8 F= -.2888D+05 Step=  .4820D+00 Slope= -.5147D+03
Try =  9 F= -.2889D+05 Step=  .4937D+00 Slope= -.5230D+03
Try = 10 F= -.2889D+05 Step=  .5039D+00 Slope= -.5304D+03
Try = 11 F= -.2890D+05 Step=  .5131D+00 Slope= -.5373D+03
Try = 12 F= -.2890D+05 Step=  .5214D+00 Slope= -.5437D+03
Try = 13 F= -.2891D+05 Step=  .5290D+00 Slope= -.5497D+03
Try = 14 F= -.2891D+05 Step=  .5360D+00 Slope= -.5553D+03
Try = 15 F= -.2892D+05 Step=  .5425D+00 Slope= -.5607D+03
Try = 16 F= -.2892D+05 Step=  .5486D+00 Slope= -.5658D+03
Try = 17 F= -.2892D+05 Step=  .5542D+00 Slope= -.5706D+03
Try = 18 F= -.2892D+05 Step=  .5596D+00 Slope= -.5752D+03
Try = 19 F= -.2893D+05 Step=  .5646D+00 Slope=  .2016D+03
Try = 20 F= -.2893D+05 Step=  .5642D+00 Slope= -.5793D+03
1st derivs.      .74139D+01   .55601D+03   .16343D+03
Parameters:      .99292D+00   .46368D+00   .82497D+00
Itr  2 F= -.2893D+05 gtHg= .5796D+03 chg.F=.2365D+03 max|db|=  .1199D+04
Try =  0 F= -.2893D+05 Step=  .0000D+00 Slope= -.5796D+03
Try =  1 F= -.2861D+05 Step=  .1000D+00 Slope=  .7339D+04
Try =  2 F= -.2893D+05 Step=  .8642D-02 Slope=  .1883D+03
Try =  3 F= -.2893D+05 Step=  .1960D-02 Slope=  .2047D+03
Try =  4 F= -.2893D+05 Step=  .4751D-03 Slope=  .2083D+03
Try =  5 F= -.2893D+05 Step=  .1647D-03 Slope=  .2090D+03
Try =  6 F= -.2893D+05 Step=  .1372D-03 Slope=  .1043D+03
Try =  7 F= -.2893D+05 Step=  .1274D-03 Slope=  .2392D+02
1st derivs.    -.12633D+02  -.72807D+02   .16345D+03
Parameters:     .99292D+00   .46356D+00   .82494D+00
Itr  3 F= -.2893D+05 gtHg=  .1794D+03 chg.F=  .5362D-01 max|db|=  .1981D+03
1st derivs.    -.12633D+02  -.72807D+02   .16345D+03
Parameters:     .99292D+00   .46356D+00   .82494D+00
Itr  1 F= -.2893D+05 gtHg=  .1794D+03 chg.F=  .2893D+05 max|db|=  .1981D+03
Try =  0 F= -.2893D+05 Step=  .0000D+00 Slope= -.1794D+03
Try =  1 F= -.2893D+05 Step=  .1274D-03 Slope=  .8810D+01
1st derivs.     .74150D+01   .38739D+03   .16346D+03
Parameters:     .99293D+00   .46361D+00   .82482D+00
Itr  2 F= -.2893D+05 gtHg=  .4008D+03 chg.F=  .1036D-01 max|db|=  .1187D+04
Try =  0 F= -.2893D+05 Step=  .0000D+00 Slope= -.1637D+03
Try =  1 F= -.2893D+05 Step=  .1274D-03 Slope= -.1637D+03
Try =  2 F= -.2893D+05 Step=  .2547D-03 Slope= -.1637D+03
Try =  3 F= -.2893D+05 Step=  .5095D-03 Slope= -.1638D+03
Try =  4 F= -.2893D+05 Step=  .1019D-02 Slope= -.1638D+03
Try =  5 F= -.2893D+05 Step=  .2038D-02 Slope= -.1639D+03
Try =  6 F= -.2893D+05 Step=  .4076D-02 Slope= -.1640D+03
Try =  7 F= -.2893D+05 Step=  .8151D-02 Slope= -.1643D+03
Try =  8 F= -.2893D+05 Step=  .1630D-01 Slope= -.1649D+03
Try =  9 F= -.2893D+05 Step=  .3261D-01 Slope= -.1673D+03
Try = 10 F= -.2894D+05 Step=  .6521D-01 Slope= -.1698D+03
Try = 11 F= -.2895D+05 Step=  .1304D+00 Slope= -.1769D+03
Try = 12 F= -.2897D+05 Step=  .2608D+00 Slope= -.1933D+03
Try = 13 F= -.2903D+05 Step=  .5217D+00 Slope= -.2384D+03
Try = 14 F= -.2849D+05 Step=  .1043D+01 Slope=  .2398D+05
Try = 15 F= -.2912D+05 Step=  .8520D+00 Slope= -.3463D+03
Try = 16 F= -.2916D+05 Step=  .9545D+00 Slope= -.4082D+03
Try = 17 F= -.2915D+05 Step=  .9673D+00 Slope=  .2080D+04
Try = 18 F= -.2916D+05 Step=  .9556D+00 Slope=  .8732D+03
Try = 19 F= -.2916D+05 Step=  .9553D+00 Slope= -.4088D+03
Try = 20 F= -.2916D+05 Step=  .9554D+00 Slope= -.4089D+03
1st derivs.    -.76681D+01   .13325D+03   .40881D+03
Parameters:     .10480D+01   .46122D+00  -.12901D+00
Itr  3 F= -.2916D+05 gtHg=  .1001D+04 chg.F=  .2351D+03 max|db|=  .1896D+05
Try =  0 F= -.2916D+05 Step=  .0000D+00 Slope= -.4089D+03
Try =  1 F= -.2916D+05 Step=  .1274D-03 Slope=  .8737D+03
Try =  2 F= -.2916D+05 Step=  .3105D-04 Slope= -.4084D+03
Try =  3 F= -.2916D+05 Step=  .4441D-04 Slope= -.2954D+03
Try =  4 F= -.2916D+05 Step=  .5146D-04 Slope=  .5476D+02
Try =  5 F= -.2916D+05 Step=  .5127D-04 Slope=  .4534D+02
Try =  6 F= -.2916D+05 Step=  .5112D-04 Slope=  .3764D+02
1st derivs.    -.76674D+01  -.40340D+03  -.37166D+02
Parameters:     .10480D+01   .46122D+00  -.12906D+00
Line search does not improve fn. Exit iterations. Status=3
Abnormal exit from iterations. If current results are shown
check convergence values shown below. This may not be a
solution value (especially if initial iterations stopped).
Gradient value: Tolerance= .1000D-05, current value= .4842D+03
Function chg. : Tolerance= .0000D+00, current value= .1337D-06
Parameters chg: Tolerance= .0000D+00, current value= .1462D+04
Smallest abs. parameter change from start value = .4793D-01
Function= -.28691160722D+05, at entry, -.29162843696D+05 at exit
```

R26.4.4 Too Few Iterations

LIMDEP is generally not able to discern if something is wrong with your estimation problem or, of course, if the specification of your model is incorrect. Note in the preceding example, the failure of the iterations is noted with a message about not finding the function optimum (the problem was constructed so that the function had no optimum), followed by a suggestion that something *appears* to be wrong. Another possible iteration failure is diagnosed when what looks like convergence occurs more quickly than might be expected. Thus, if you fit a model that looks 'too good,' you might get something like the following message:

```
Itr  2 F=  .3465D+04 gtHg=.2434D-02 chg.F=.6227D-06 max|db|=.8534D+00
Try =  0 F=  .3465D+04 Step=  .0000D+00 Slope= -.2432D-02
Try =  1 F=  .3465D+04 Step=  .1973D-04 Slope=  .6015D-01
Try =  2 F=  .3465D+04 Step=  .7670D-06 Slope=  .1673D-09
1st derivs.     .51135D-06  -.43989D-05   .52469D-05  -.82815D-06
Parameters:    -.26491D-02   .50561D-02  -.15405D-01   .26967D-02
Itr  3 F=  .3465D+04 gtHg=.6916D-05 chg.F=.9345D-09 max|db|=.8725D-03
Try =  0 F=  .3465D+04 Step=  .0000D+00 Slope= -.6916D-05
Try =  1 F=  .3465D+04 Step=  .7670D-06 Slope=  .2358D-02
Try =  2 F=  .3465D+04 Step=  .2243D-08 Slope=  .7125D-13
1st derivs.     .33793D-07   .79251D-07   .62163D-07  -.89005D-08
Parameters:    -.26491D-02   .50561D-02  -.15405D-01   .26967D-02
Itr  4 F=  .3465D+04 gtHg=.9579D-07 chg.F=.1501D-10 max|db|=.1774D-04
                        * Converged
Note: DFP and BFGS usually take more than 4 or 5 iterations to converge.
If this problem was not structured for quick convergence, you might want
to examine results closely. If convergence is too early, tighten convergence.
Normal exit from iterations. Exit status=0.
```

This is not necessarily an error message, but it might indicate a problem. For example, if you restart an estimation problem using as starting values the values which maximize the function already, then convergence will come very quickly. The particular model which generated the message above was extremely well behaved, and it did converge to a true optimum in only three iterations. But, that is relatively unusual, so when you see this message, you should check to insure that the convergence was to a true optimum of your function.

R26.4.5 General Failure of Indeterminate Cause

Finally, another possibility is that apparent convergence of the estimator isn't convergence at all. The following message was produced by another deliberately badly structured estimation problem. For the particular function we chose, the first derivatives are always identically zero at the starting values, but that is not a maximum of the function. Because zero derivatives is a convergence rule, it looks to the optimizer as if an optimum has been found in the first iteration. But, that is definitely not the case. The signature of this failure is the last line of the output, which states that the estimated covariance matrix of the estimates is singular. In this situation, the 'estimates' are probably nonsense values.

```
NOTE: Convergence in initial iterations is rarely
at a true function optimum.  Check all results.
check convergence values shown below. This may not be a
solution value (especially if initial iterations stopped).
Gradient value: Tolerance= .1000D-05, current value= .0000D+00
Function chg. : Tolerance= .0000D+00, current value= .3059D+05
Parameters chg: Tolerance= .0000D+00, current value= .0000D+00
Smallest abs. parameter change from start value = .0000D+00
Note:  At least one parameter did not leave start value.
Normal exit from iterations. Exit status=0.
Models - estimated variance matrix of estimates is singular
Current estimated covariance matrix for slopes is singular.
```

R26.4.6 Interrupting the Iterations

You may find it necessary to stop the iterations. For example, it may become obvious that something is wrong, and this estimation process is not going to work. When this occurs, especially if you are using a very large data set, you can stop the iterations by clicking the red Stop button on the *LIMDEP* toolbar, shown in Figure R26.2.

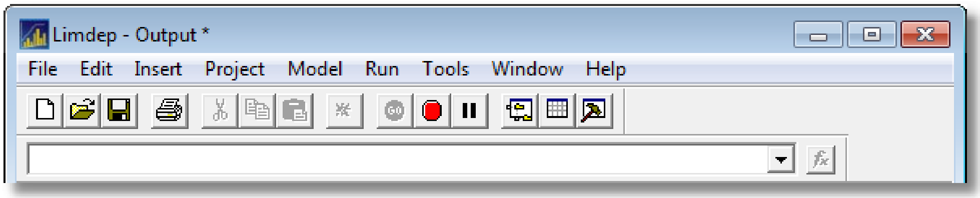
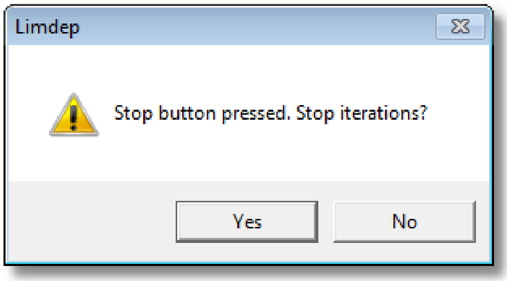


Figure R26.2 Stop Button on the *LIMDEP* Toolbar

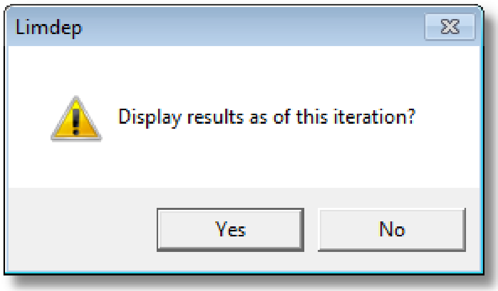
The Stop button operates as a yes/no query. During iterations or other operations that iterate by looping over the data, the Stop button on the toolbar will turn red. If you click this button during computation, *LIMDEP* will interrupt the computations and ask you if you would like to end the iterations at this point, as shown in Figure R26.3. The query appears after a full pass through the data set is complete. If you have a very large data set, there may be a noticeable lag between when you click the Stop button and the dialog box appears.


Limdep

Stop button pressed. Stop iterations?

Figure R26.3 Query for Exit from Iterations

If you elect to stop the iterations and enough iterations have been carried out that some progress has been made toward a solution, you will be offered a display of the results as they have been obtained up to that point. This is shown in Figure R26.4



Display results as of this iteration?

Figure R26.4 Exit from Iterations to Current Results

The toolbar also contains a Pause button to the right of the Stop button. You can click this to interrupt an ongoing procedure, then click it again to continue.

R26.4.7 Warnings During the Iterations

Certain warnings do occur during optimization, such as for a correlation that strays out of the [-1.1] interval. These warnings only indicate that a trial value of the parameter was not a valid estimate. *LIMDEP* will then back up and try a new value, and the iterations will continue. So, you will often see these diagnostics interspersed with other output.

NOTE: These warnings should be ignored if the estimator subsequently reaches a normal convergence. If convergence is not subsequently reached, the warnings may help you diagnose the problem.

R26.5 Exit Codes

All estimation procedures produce a scalar named *exitcode*. The exit status from any model routine will be shown in your trace file, trace.lim, or in your output file if you were using one. The exit codes that are reported are

0. Normal exit, everything OK. The convergence rule that was satisfied will be listed.
1. Maximum iterations exceeded.
2. Failure, singular Hessian.
3. Failure, unable to maximize function.
4. Unable to compute function value. (This will be rare. Bad starting values will cause this.)
5. General failure at setup time, not during estimation. Another diagnostic will appear i n the trace file to explain this. (Almost always an error in your command or data.)

The *exitcode* is accessible as any other scalar, and can be used for any desired purpose. For example, if you have written a program that fits models in a loop, you may want to skip certain computations if *exitcode* is not 0, since in this case, a certain value that you might want to retrieve may be unavailable. There is a mixture of termination conditions in *exitcode*, both for model setup and for the optimization, itself. A second indicator is produced by the nonlinear optimization, named *opt_exit*, which gives only the information 0 - 4 above, and only during optimization. This code is a somewhat better indicator of the actual optimization process.

R26.6 Iteration Controls

You can control several aspects of the iterations with options that can appear in your commands.

R26.6.1 Maximum Iterations

To control the maximum number of iterations taken by the iterative routines use

; Maxit = maximum

The defaults are 50 for the algorithms which use a line search (DFP, BFGS, and BHHH) and 25 for Newton's method.

NOTE: The special case, **; Maxit = 0** is used to carry out LM tests based on the starting values that you provide. **; Maxit = 0** is a specific instruction to do this test. See Chapter R13 for details.
TIP: If your estimator goes through many iterations, then fails with a singular Hessian after apparently converging, reestimate the model, and set Maxit to some value lower than the number reached on the previous try. You may learn something useful about the model this way.

R26.6.2 Algorithms

All models have a default algorithm. We have chosen the one most likely to work in most cases. (See the discussion below.) In most cases, including *LIMDEP*'s minimization package, an alternative algorithm is requested in a command with the specification

; Alg = algorithm

The chapters to follow which describe the models in detail list the defaults and available options for *LIMDEP*'s nonlinear models. Also shown is the method used to compute the estimate of the covariance matrix of the coefficients, which is sometimes part of the algorithm. The choices are

DFP	Davidon, Fletcher, Powell
BFGS	Broyden, Fletcher, Goldfarb, Shanno
NEWTON	Newton's Method
BHHH	Berndt, Hall, Hall, Hausman
SteDes	Steepest Descent

Technical material on these methods appears below. You may choose an algorithm by its first two letters.

It should be noted, unless the convergence criteria are made fairly tight, the different algorithms will often give slightly different answers (i.e., at the fifth or sixth significant digit). Which is likely to be most successful is going to depend on the data, and it may pay to try more than one. The defaults chosen have been found to be the most reliable. Generally, BFGS is most likely to be successful when no particular choice is obvious. However, it can be rather time consuming. In a very large data set, use Newton or BHHH, if possible, BFGS or DFP if they fail to find the maximum.

Newton’s method is best if the problem is globally concave, but this is somewhat unusual. Our experience has been that using Newton’s method when the BHHH estimator is used to estimate the covariance matrix often fails to converge. For example, **; Alg = BHHH** will probably not work very well for the bivariate probit model, but might be satisfactory for the logit model. On the other hand, for the logit model, Newton’s method will almost always be the best of the group, as the log likelihood for this model is globally concave. Steepest descent is almost always extremely slow to converge and will often fail altogether. We advise against using it at all.

R26.6.3 Convergence Rules

Convergence can be based on any of three criteria:

- **Gradient:** $\mathbf{g'Hg} < \varepsilon_g$ where **g** is the current derivative vector and **H** is the inverse of the current Hessian or, in the case of DFP and BFGS, the most recent estimate of it. This value is reported as ‘Derivative’ in the technical display and `gtHg` in the iteration output.

- **Proportional change in all parameters** $< \varepsilon_b$.

- **Proportional change in the function value** $< \varepsilon_f$.

The default value of ε_g is 0.00001 and the other two are set to 0.0. By default, *LIMDEP* uses only the gradient rule. (This is the standard choice in contemporary software.) In general, convergence will be based on any nonzero value for these three rules.

You can change the gradient value and/or activate the other two as described below. Note in particular the gradient criterion. This is not based on the absolute size of the derivatives, as they are dependent on the scale of the data. Rather, the criterion is based on a scale free quadratic form. As such, an iteration may converge at a point at which the derivatives are a bit larger than you might have expected.

To change the values of these settings for a particular model, use

; Tlf	=	**value**	to set the function convergence criterion,
; Tlb	=	**value**	to set the parameter convergence criterion,
; Tlg	=	**value**	to set the gradient convergence criterion.

If you omit the ‘= value,’ 0.0 is assumed. These apply only to the model command in which they appear. On the next model, they will again be set at the default values.

For practical purposes, the gradient rule is generally the best of the three. The least satisfactory will usually be the parameter rule. Convergence on the function might be useful for a particularly difficult problem, but it should be noted that the gradient rule is generally more difficult to satisfy. As such, if you have both Tlf and Tlg turned on, Tlg will often be reached first, and thus, will often result in fewer iterations to convergence.

NOTE: When you do reach convergence on the parameters or the function, you should check the derivative rule anyway. It is possible for $\mathbf{g'Hg}$ to be too large, even if convergence on the function value seems to have been reached.

R26.7 Quadrature

Several estimators in *LIMDEP* use Gaussian quadrature to approximate integrals that cannot be evaluated analytically. We use Hermite quadrature when the limits are -∞ to +∞ and Laguerre quadrature when the limits are 0 to ∞. The computation is

$$\int_0^\infty e^{-x} f(x)dx \approx \sum_{i=1}^{L} w_i f(z_i)$$

for Gauss-Laguerre quadrature and likewise for Gauss-Hermite quadrature in which the weight function is exp($-x^2$) and the range of integration is (-∞,∞). The values w_i are the 'weights' for the quadrature and z_i are the 'nodes.' The approximations differ in accuracy based on the number of points used. The more points are used, the more accurate the approximation, but, at the same time, the greater is the amount of time needed to do the computation. Gaussian quadrature is described in detail in Abramovitz and Stegun (1972). (We do note, the approximations are surprisingly accurate even for fairly small numbers of nodes.)

At the time you start *LIMDEP*, the default numbers of points for these quadratures are 20 and 40 points respectively. This uses an intermediate value for Hermite quadrature; 68 points are provided for Laguerre and 96 for Hermite if you wish to choose a higher setting. More points provide greater accuracy, so all else constant, these choices are optimal. But, the greater number of points also increases the amount of computation time, so you may want to reduce these values.

Any model command may contain

; Lpt = number of points for Laguerre quadrature, one of
2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 20, 40, 68

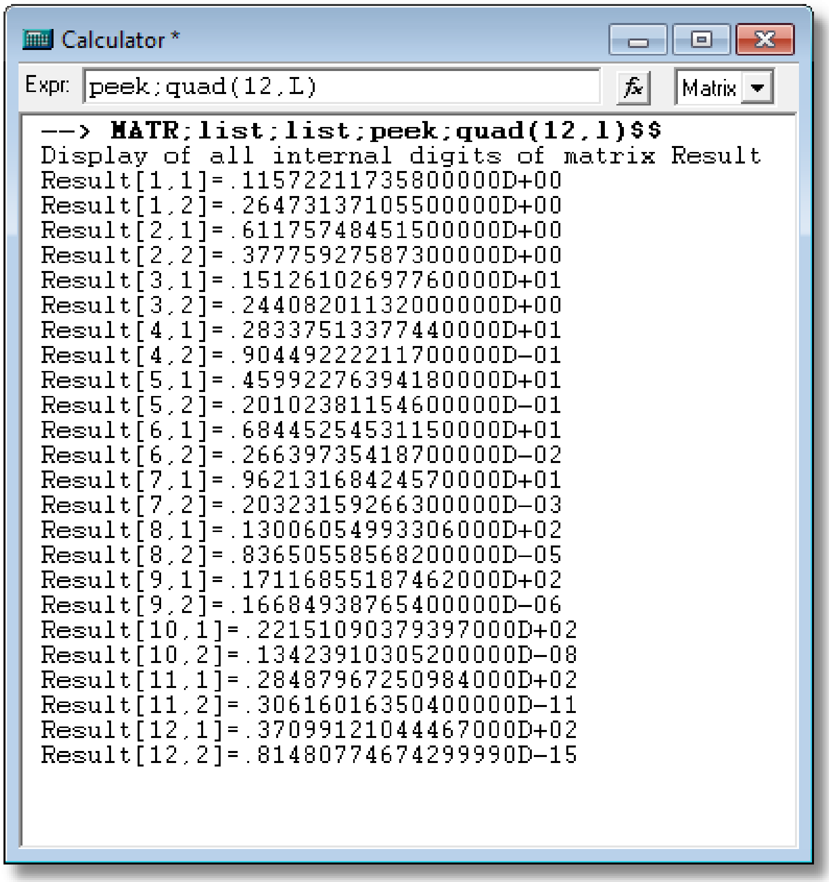
; Hpt = number of points for Hermite quadrature, one of
2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 16, 20, 32, 64, 96

(*LIMDEP* will choose the closest value it can to the value you specify if your value is not in the list.) The model need not actually use these techniques. Once the number of quadrature points is set by a model command, it remains set until another model command changes it.

You might find it useful to have access directly to these vectors of weights and abscissas for example, to approximate an integral of your own specification. A **MATRIX** command is provided for this purpose;

MATRIX ; [name =] Quad (number of points, L or H) $

(The 'name =' is optional. **MATRIX** is discussed further in Chapter R16.) An example is shown in Figure R26.5



```
peek;quad(12,L)
```

```
--> MATR;list;list;peek;quad(12,1)$$
Display of all internal digits of matrix Result
Result[1,1]=.115722117358000000D+00
Result[1,2]=.264731371055000000D+00
Result[2,1]=.611757484515000000D+00
Result[2,2]=.377759275873000000D+00
Result[3,1]=.151261026977600000D+01
Result[3,2]=.244082011320000000D+00
Result[4,1]=.283375133774400000D+01
Result[4,2]=.904492222117000000D-01
Result[5,1]=.459922763941800000D+01
Result[5,2]=.201023811546000000D-01
Result[6,1]=.684452545311500000D+01
Result[6,2]=.266397354187000000D-02
Result[7,1]=.962131684245700000D+01
Result[7,2]=.203231592663000000D-03
Result[8,1]=.130060549933060000D+02
Result[8,2]=.836505585682000000D-05
Result[9,1]=.171168551874620000D+02
Result[9,2]=.166849387654000000D-06
Result[10,1]=.221510903793970000D+02
Result[10,2]=.134239103052000000D-08
Result[11,1]=.284879672509840000D+02
Result[11,2]=.306160163504000000D-11
Result[12,1]=.370991210444670000D+02
Result[12,2]=.814807746742999990D-15
```

Figure R26.5 MATRIX Function to Retrieve Quadrature Weights

R26.8 Multivariate Normal Probabilities

LIMDEP contains a simulator for the multivariate normal CDF. We use the GHK (Geweke, Hajivassiliou, Keane) methodology to approximate the CDF. (See Greene (2012) for details.) The technique produces quite fast and accurate approximations to the *M* fold integral

$$P = \int_{A(M)}^{B(M)} \dots \int_{A(1)}^{B(1)} f(x_1,\dots,x_M)dx_1,,,dx_M.$$

where *f*(...) is the *M*-variate normal density function for *x* with mean vector zero and *M*×*M* positive definite covariance matrix, **Ω**. The approximation is obtained by averaging a set of *R* replications obtained by transforming draws produced by a random number generator. (See Appendix R4A.2 for details on the computation.) The simulation estimator of *P* is consistent in *R*. Further details may be found in Greene (2012) and in a symposium in the November, 1994 *Review of Economics and Statistics* and the references cited there. Usage, including how to set *R* is discussed below. *M* may be up to 20, though the accuracy for a given *R* declines with *M*. For any *M*, the accuracy increases with *R*. Again, the estimated *P* is consistent in *R*.

We have implemented this procedure in four applications in *LIMDEP* –**CALCULATE, CREATE, MATRIX**, and a multivariate probit model which extends the bivariate probit model – as well as the multinomial probit extension of the logit model available in *NLOGIT* Version 6. Users of this technique should be familiar with the theoretical development, including its limitations, as discussed in the received literature.

The value of *R*, the number of replications, is set globally, at the time you start *LIMDEP*, at 100. Authors differ on how large *R* must be to get good approximations. The default 100 is a compromise. Some have mentioned 500. You may change *R*, but be aware that higher *R* leads to greatly increased amounts of computation; estimators which use this technique are slow. Two ways to set *R* are with **CALC** and in the estimation commands. First,

CALC ; Rep(r) $ for example, **CALC ; Rep(100) $**

sets *R* permanently. The model command which uses the simulator is the multivariate (multiple equation) probit model. The command can include a setting for *R*, which is then permanent until changed later, with

MPROBIT ; ... ; Pts = r ; ... $

R26.8.1 Model Based on the Multivariate Normal Distribution

The model which use this technique is the multivariate probit model. The multivariate probit model is a multiequation extension of the probit model;

$$y_{im}^* = \boldsymbol{\beta}_m'\mathbf{x}_{im} + \varepsilon_{im}$$
$$y_{im} = 1 \text{ if } y_{im}^* > 0\text{, and 0 otherwise, } m = 1,...,M.$$

The ε_{im}, $m = 1,...,M$ have a multivariate normal distribution with variances 1 and correlations ρ_{ml}. Each individual equation is a standard probit model. This generalizes the bivariate probit model for up to 20 equations. Note the difference between the *multivariate* probit model and the *multinomial* probit model. The multivariate probit model defines the outcomes of up to 20 dependent variables. All of them obey their own behavioral equation. In the multinomial probit model, there is a single dependent variable and a single outcome. This is described in greater detail in the *NLOGIT Reference Guide*

R26.8.2 Tools that Calculate Multivariate Normal Probabilities

For computing multivariate normal probabilities to be used in scalars, variables, or matrix results, you can use the following:

For single computations of the multivariate normal CDF in which $A_m = -\infty$, you may use

CALCULATE ; Result = Mvn(x,w) $

to obtain a single probability from $-\infty$ to x_m, $m = 1,...,M$. The parameters *x* and *w* are a vector and a matrix, respectively, obtained using **MATRIX** or as the result of some estimation procedure, where *x*(.) provides the upper bounds.

If you desire to compute the probability in a rectangle defined by finite *A*(.) at the lower limits and *x*(.) at the upper limits, use

```
CALCULATE ; Result = Mvn(x,w,a) $
```

If you desire complementary probabilities, that is the probability for the area defined by a lower bound of *x*(.) and upper bounds of +∞, use Mvn(*y*,*w*) where *y* is the negative of *x*. You may also obtain the multivariate normal density, with the function

```
CALC          ; Result = Mvd(x,w) $
```

Further details on calculating multivariate normal probabilities appear in Section R17.6.7.

You may use

```
CREATE        ; Prob = Mvn(x,w) $
```

in which *x* is a namelist of *M* variables. Each row (observation) in *x* is the counterpart to the *x* in the **CALCULATE** function. As before, *x* is *M* variables and *w* is the *M*×*M* covariance matrix. Note, variables may be repeated in *x*. For example, if *x*1 and *x*2 are free, but *x*3 - *x*6 are all zero, then you could use

```
CREATE        ; zero = 0 $
NAMELIST      ; x = x1,x2,zero,zero,zero $
CREATE        ; p = Mvn(x,w) $
```

This creates a variable *p* with each element equal to the *M*-variate normal CDF evaluated at *w* and the *i*th observation in *x*. The Mvn function may be used as you would any other function in **CREATE**. The function Mvd(*x*,*w*) returns the column vector of densities instead of the CDF.

You may use

```
MATRIX        ; Result = Mvnp(x,w) $
```

Once again, *w* is the *M*×*M* covariance matrix. Now, *x* is an *N*×*M* matrix or a namelist of *M* variables. The result is an *N*×1 column vector of the normal probabilities. This operates essentially the same as the **CREATE** function – the differences are that *x* may be a conformable matrix or a namelist and the result of the computation is a matrix, not a variable. The function

```
MATRIX        ; Result = Mvnd(x,w) $
```

returns the density functions instead of cumulative probabilities.

R26.9 Default Values of Program Parameters

As described above, there are a set of program default settings for optimization parameters. These are

Program Parameter	Default	Command Setting
Convergence Rules		
Parameters	0.0	**; Tlb = value** (0.0 to 1.d-8)
Function	0.0	**; Tlf = value** (0.0 to 1.d-8)
Gradient	1.d-6	**; Tlg = value** (0.0 to 1.d-8)
Quadrature Points		
Hermite	40	**; Hpt = number** (2 to 96)
Laguerre	20	**; Lpt = number** (2 to 68)
Iterations		
Iterations	100	**; Maxit = number** (0 to 1000)
Technical Output	0	**; Output = value** (0,1,2,3,4,5)
Simulation points	100	**; Pts = value** (1 to 2000)
Model Output		
Confidence Level	.95	**; CL = value** (0.1 to 0.995)
Information Criteria	0	**; IC = value** (0 or 1)

You may change any of these in a specific model command. When you do so, the setting is for that model only. After estimation, the values return to their default setting. You may change the default settings by using

DEFAULT ; parameter setting(s) as shown above **$**

The command may change any or all of the defaults. These will remain in effect going forward from that point until you change them again. Note, however, these settings are now written into a project file, so if you exit then restart *LIMDEP*, the program defaults will be their original settings.
You can obtain a listing of the current default settings with

DEFAULT $

The listing will appear as shown below.

```
Current Settings of Program Defaults for Estimation
---------------------------------------------------
Convergence criteria for optimization program
            Change in function       .0000000
            Change in parameters     .0000000
            Derivative criterion     .0000010
Maximum iterations                        100
Technical output during iterations          0
Information criteria beyond AIC             0
Hermite quadrature points                  40
Gauss Laguerre quadrature points           20
Number of draws for simulations           100
Confidence level in confidence intervals  95%
Maximum utility in multinomial choice   100.0
---------------------------------------------------
```

R26.10 Starting Values

When you wish to provide your own starting values for any nonlinear model, you will add the specification

; Start = list of values

to your model command. If starting values are not provided, the program will usually use least squares, or some variant, or perhaps zero.

TIP: With a few exceptions including the **MAXIMIZE/MINIMIZE** feature and related commands in which you specify your own model, and the bivariate ordered probit model, starting values are *always* optional.

The chapters on the specific models will show exactly what is required if you wish to provide starting values. If you do provide starting values, you must usually give a complete set, including any ancillary parameters, such as the disturbance standard deviation, σ, for the tobit model. For example, the ML estimator for the selection model requires starting values for α (probit), β (regression), σ, and ρ. Your command might look as follows:

```
SELECT      ; Lhs = y ; Rhs = x
            ; MLE
            ; Start = alpha, beta, sigma11, -0.321 $
```

where *alpha* and *beta* are matrices (vectors), *sigma*11 is a scalar, and the value for ρ is given explicitly. In any list, you may give values as particular numbers, in calculator scalars, or in matrices or any mixture as long as the right number of values, in the right order, are given in total. A convenient device is the repetition factor for multiple occurrences of the same value,

K_value = value,value,...,value *K* times.

The repetition value may be a literal number or a scalar. For example, a general routine for starting the iterations for a probit model at [**0**] would be:

```
NAMELIST    ; x = ... the list of Rhs variables $
CALC        ; k = Col(x) $
PROBIT      ; Lhs = y ; Rhs = x
            ; Start = k_0 $
```

R26.11 Hints for Iterative Estimation

When you give a model command, chances are good that estimation will proceed smoothly and, eventually, your estimates will be reported after some intermediate output which may or may not be of interest to you. But, there are a number of things that can go wrong with a nonlinear estimator. The following lists some of the most common problems and how you can react to them when they occur.

Model Development

The paragraphs below will detail several possible problems. There are many ways that estimation of a nonlinear model can fail, and we cannot anticipate all of them here. One piece of advice that we give more than any other is the following: If estimation of a model breaks down for nonobvious reasons, it is often related to the data. Back up and try fitting an extremely simple version of the same model with perhaps only one or two variables in it. This is not intended to be a specification that would be interesting. But, starting from a very simple formulation will usually allow you to get the estimation process started. Then, add in variables one or two at a time. At some point, presumably before the initial failure occurred, the estimation will break down, revealing to you the variable that is causing the problem.

Scaling Data

This is often crucial. If you are having convergence problems, the first place to look is at the scaling of your data. Models such as the ordered probit will often fail to converge if your model contains variables of very different magnitudes. For example, a model which includes dummy variables (order 1), as well as income in dollars (order 10,000) and income squared (order 100,000,000) will almost never be estimable. All you have to do is divide income by 10,000 before squaring, and use scaled income and income squared. In many cases, this is all that is needed to turn an ill behaved problem into a well behaved one.

One of the indicators of a scaling problem, aside from the statistics for the variables themselves, is badly unbalanced derivatives. If you suspect this problem, try doing the estimation with an output file open. After exiting from the iterations, inspect the file. The output will list, for each iteration, the values of the parameters and the derivatives. If you find that one or a few of the derivatives are very much larger than the others, say 1,000 compared to 1 or .1, then the variable associated with that coefficient needs to be scaled.

Model Size

LIMDEP makes it very easy to specify very large models with very little effort, once the data are read. If you are estimating one of the ‘messier’ models, such as the bivariate probit model or the mover stayer model – these are models with very volatile log likelihoods which are difficult to maximize – you may find it useful to begin your analysis with a very small model, even if the specification is suspect, just to get started. Choose a small handful of variables, say two or three at the most, and estimate the model. If you find that it works, start building up the model by adding back the variables you think should be in the equation(s). If the procedure should break down somewhere along the way, you will be able to single out problematic variables. On the other hand, if the procedure breaks down at the very first step, you should give some thought to whether your specification is correct. A mismatch between the model and the dependent variable may be the issue.

Warnings About Parameter Values

Many of *LIMDEP*'s models contain parameters which are restricted to a particular range. For example, the correlation coefficient which appears in the sample selection, switching regressions, and bivariate probit models must be in the range -1 to 1. The threshold parameters in the ordered probit model must be strictly increasing. Models which contain such parameters are estimated by the BFGS method, with its elaborate line search. So, from time to time you may get a diagnostic that, for example, an estimate of a variance parameter is out of range (negative). There is nothing for you to do, and nothing is wrong. *LIMDEP* has simply tried out a value of the parameter vector which has an invalid value. The diagnostic is triggered, and *LIMDEP* will try a smaller step size. The step size will be reduced until the value is no longer invalid. (It is certain that the step size can be reduced enough to produce a valid value; the value from which the steps began must have been valid.)

On the other hand, if the parameter values are invalid at the very beginning of the iterations, *LIMDEP* will give up immediately. Internally, *LIMDEP* makes sure that starting values are always valid. But, you may supply your own values, and if you send in a bad value, rather than attempting to patch up the error, *LIMDEP* quits and waits for you to respecify the model.

Premature Convergence

Be suspicious of convergence which occurs when the derivatives are large. The function value may be relatively stable even though the derivatives are not close enough to zero if your model is not very well specified and as a consequence, *LIMDEP* is taking very small steps from one iteration to the next. The values of the derivatives of the function at a 'true' convergence are typically from 1.d-2 to 1.d-6. If you have convergence at values in the range of 1.d+2 or so, you may want to reestimate the model with the function convergence switch turned off with **; Tlf**. Now, the opposite may also occur. For your data, it may simply be impossible to find a satisfactory point at which the derivatives are small. Rather than let *LIMDEP* give up, you may want to force convergence, which you can also do with **; Tlf**, using a relatively large value, say .01. This takes some experimenting. We should note, your best approach would be to leave the convergence rules as the program sets them until problems arise.

Appendix R26A Technical Details on Optimization

Most of the nonlinear optimization programs in *LIMDEP* use what are called gradient methods to maximize log likelihoods. The class of gradient methods is defined by the iteration

$$\mathbf{b}_{t+1} = \mathbf{b}_t - \lambda_t \mathbf{H}_t \mathbf{g}_t$$

where $\mathbf{b}_{t+1}$ = the next value of the estimate,
$\mathbf{b}_t$ = the current value of the estimate,
λ_t = the stepsize,
$\mathbf{g}_t$ = the vector of first derivatives, or gradient,
$\mathbf{H}_t$ = a positive definite matrix.

The subscript '*t*' indicates that the quantity is computed using $\mathbf{b}_t$. The iterative procedure begins with a 'starting value,' $\mathbf{b}_0$. At each iteration, the gradient and **H** are computed, a decision is made as to what is the best stepsize to take, and the next value of **b** is computed. At some point, either because of 'convergence,' too many iterations, or failure of the procedure (by its own determination) to locate the optimal parameter value, the iterative procedure is ended and the results are reported. *Convergence* is deemed to occur when one or more stopping rules are met. Because of rounding and approximation error, the search for the best **b** must end before the derivatives of the log likelihood are exactly zero. Thus, one or more *convergence criteria* are defined to determine when the estimates are 'close enough' to the true maximizers of the log likelihood to quit.

During its iterations, *LIMDEP* is *minimizing* the negative of the log likelihood. As such, when you see function values in your output during iterations, they will have the opposite sign of the log likelihood that appears in your output. In addition, there are a few cases, such as the constant $-(N/2)\log(2\pi)$ terms in log likelihood functions built up from the normal density, in which *LIMDEP* does not bother with invariant constants during its iterations. In these cases, the log likelihood function reported to you in your final results may not equal the minimized value of the function that was minimized to obtain the estimates.

An 'algorithm' is defined by the particular choice of λ and **H**. *LIMDEP* offers several, but chooses a 'default' for itself unless you specifically change the choice. A 'line search' is rather like an iteration within an iteration. The line search consists of a search for the best stepsize for a given **H** and **g** in a given iteration.

LIMDEP generally uses one of two algorithms for solving the nonlinear optimization problems. For globally convex problems, such as, tobit, logit and probit, Newton's method is a natural choice. For details on the algorithm, see, e.g., Greene (2012). For the less well behaved likelihood functions, a form of the method of Broyden et. al (see Fletcher, 1980) is used. The basic algorithm is also described in Greene (2012). *LIMDEP* uses a modification of this method suggested by Gruvaeus and Joreskog (1970). The algorithm uses a line search method developed by the aforementioned authors in order to determine a stepsize. This is coupled with the formulation of the BFGS algorithm as described, e.g., in Greene (2012). The search procedure begins with several steepest descent iterations (using the same line search) in order to improve the starting values. It then continues with the BFGS iterations. Thus, when you see your intermediate output, you will see the parameter search in two parts with what appears to be two consecutive sets of iterations.

Alternative algorithms may be chosen for fitting most of *LIMDEP*'s nonlinear models. The following are the algorithms as defined by *LIMDEP*. In a few cases, these differ slightly from the conventional usage. In each case, **H** is an estimate of the asymptotic covariance matrix of the coefficient estimates, or the inverse of the Hessian. This may use analytic second derivatives or the sum of the outer products of the first derivatives depending on the model. The gradient in all cases is denoted '**g**.' Where it appears, λ is the step length found by the line search. The number of the iteration is denoted '*t*'.

Newton's Method: $\mathbf{b}_{t+1} = \mathbf{b}_t - \mathbf{H}_t\mathbf{g}_t$.

Most of the routines that use this method use actual second derivatives, not expectations. That is, we do not use the *method of scoring*. In a number of cases, however, what we call Newton's method is actually that of Berndt, et. al. without a line search. These cases are those in which we construct our estimate of **H** from first derivatives instead of the analytic second derivatives, that is, $\mathbf{H}_t = \Sigma_i \mathbf{g}_{ti}\mathbf{g}_{ti}'$, where '*i*' indexes observations.

BHHH: $\mathbf{b}_{t+1} = \mathbf{b}_t - \lambda_t\mathbf{H}_t\mathbf{g}_t$.

Save for the different method of the line search, what we call the BHHH method corresponds to the Berndt, et. al. method. But, for models such as the probit model, with its globally concave log likelihood function, what we call BHHH is actually Newton's method with a line search. (These are the cases in which **H** is built up from the analytic second derivatives rather than from the first derivatives as shown above.) The line search is that of Gruvaeus and Joreskog rather than that of Berndt, et. al.

BFGS and DFP: $\mathbf{b}_{t+1} = \mathbf{b}_t - \lambda_t\mathbf{H}_t\mathbf{g}_t$.

$\mathbf{H}_t$ is accumulated from an initial identity matrix by the rank two (DFP) or rank three (BFGS) update described in Gruvaeus and Joreskog (1970) or Greene (2012). This is used only during the iterations; the asymptotic covariance matrix is recomputed at exit from the iteration. The BFGS method is a refinement on DFP which adds another rank one term (thus making it a rank three update).

Steepest Descent: $\mathbf{b}_{t+1} = \mathbf{b}_t - \lambda_t\mathbf{g}_t$.

The steepest descent method uses only the gradient and a stepsize.

R27: Summary for *LIMDEP Reference Guide*

R27.1 Introduction

The *LIMDEP* documentation is divided into two parts. This part, the *Reference Guide*, contains descriptions of the basic program functions – data entry, file manipulation, etc. – and some suggested programs and command sets that are likely to be useful in a variety of modeling settings. The second part, the *Econometric Modeling Guide*, presents descriptions of the specific modeling frameworks that are built into the software, such as regression models, duration models, and models for binary choice. A separate *NLOGIT Reference Guide* is written specifically for users of *NLOGIT* Version 6, a suite of programs for modeling discrete choice.

This chapter will provide an overview of all of the functions in *LIMDEP*. The descriptions will include cross references to the more detailed documentations elsewhere in the manual. The purpose here is to summarize the functionality of the *LIMDEP* program suite.

R27.2 Essential Program Functions

R27.2.1 Startup

Start *LIMDEP* as you would any other program, for example, by selecting the program from the Start menu or by double clicking the *LIMDEP* icon on your desktop. (Other ways to invoke *LIMDEP* are described in Chapter R2.)

The session is identified as your 'project,' which will consist of, ultimately, your data, and the various results that you accumulate. Once you have begun a session, you will want to maintain an open editing window (the text editor) in which to enter commands. Select File:New, then Text/Command Document to open an editing window.

R27.2.2 Operation

There are two ways to give instructions to *LIMDEP*. You may use the menu driven dialog boxes (command builders), which are described Chapter R8. However, for most (probably nearly all) of the analysis you do with *LIMDEP*, the command format, with commands submitted from the text editor, is likely to be a much more convenient way to proceed. You may submit commands to the text editor in a variety of ways. You can use the Insert menu options to insert commands, the full path to a specific file, or the contents of a text file. You may execute commands using the Run menu options. Operation details are discussed in Chapter R2.

This chapter will briefly discuss the command builders and other menus. But, from this point forward, and throughout most of the *Econometric Modeling Guide*, we will rely almost entirely on the command driven mode of operation.

R27.3 Reading a Data Set

For most data sets that are stored in ASCII files, the basic instruction

IMPORT ; file = … the name of the file $

will be sufficient to import the data into *LIMDEP* ready to use. **IMPORT** and the **READ** command that can be used for unusual file types are described in Chapter R3.

The usual way to read a data file is to import variables into the data editor. You can use the menu option, Project:Import/Variables or you can open the data editor (grid icon on the toolbar) then place the mouse cursor in the empty (hatched) data field and click the right mouse button to open the data editor menu. Select Import Variables, then double click the file name to import the variables.

Once a data set has been read into your project, the variables that exist in the program will be listed in your project window. In order to view the variable listing, you will generally have to click the plus box next to the Variables topic in the Data grouping.

There are many other formats and optional specifications in **READ** that allow you to import different data sets into your data area (or into a matrix – See Section R16.5).

R27.4 Transforming Data

There are five basic commands used for data transformations:

CREATE ; variable name = expression $ to create a transformed variable,
DELETE ; list of variables $ to delete variables from the data set,
RECODE ; variable ; range of values = new value ... $ to recode a variable,
RENAME ; old name = new name $ to change the name of a variable,
SORT ; Lhs = key variable [; Rhs = variables to carry] $

You will rarely need **DELETE**, and **RENAME**, and **RECODE** should be used fairly infrequently. The **SORT** command is sometimes useful for creating index variables, but you should always remember that when you use **SORT** and you do not carry all variables, then the correspondence of variables within observations will be lost. Your data transformations will be almost exclusively carried out using **CREATE**.

CREATE is used in two ways. When you manipulate existing data, your transformations will be of the form

CREATE ; new variable = some function of existing variables $

The function on the right hand side can involve the standard mathematical operators (+, -, *, /, ^) as well as several others (! for maximum, = for creating binary variables, etc.). There are also several dozen functions, such as Log, Exp, Abs, Phi (normal CDF), and so on. The second way you will manipulate data with **CREATE** will be to generate random samples using the random number generators. In this case, your command will typically begin with

CREATE ; new variable = a column of draws from some specified distribution $

after which you will manipulate the newly created data in the usual fashion. Data transformations are described in Chapter R4. Chapter R5 details a number of other considerations for managing panel data.

R27.5 Setting the Sample

The commands used to define the sample in use at a particular time are:

NAMELIST	defines a name to be synonymous with a list of variables.
SAMPLE	designates specific observations to be included in a subsample.
REJECT	excludes certain observations from the sample based on an algebraic rule.
INCLUDE	includes certain observations from the sample based on an algebraic rule.
DRAW	draws a subsample of observations from a sample, with or without replacement.
SKIP	automatically bypasses observations that contain missing values.
DATES	establishes the periodicity of time series data.
PERIOD	designates specific time series observations to be included in a subsample.

Sample setting in its various forms is described in Chapter R7.

Prior to estimation, or during your analysis, you will want a shortcut that will enable you to equate a particular name with a group of variables. The command

NAMELIST ; name = list of variable names $

is used for this function. Namelists have several uses in model estimation, matrix algebra, and in programming estimators.

The commands

SAMPLE ; first - last $
and **PERIOD ; beginning date - ending date $**

are used to designate specific contiguous blocks of observations for inclusion in the 'current sample' for estimation purposes. The first is used for cross sections; the latter for time series. The **PERIOD** command must be preceded by

DATES ; first date in the data set $

which establishes the label of the first date in the data set and the type of data, monthly, quarterly, or yearly. The preceding are unconditional. Two commands

REJECT ; logical condition $
and **INCLUDE ; logical condition $**

are used to delete observations from or add observations to the current sample. These two commands operate on cross section data, and are generally not useful for time series data. (This is because in a time series, they would delete observations in the middle of the series, or add observations possibly randomly outside the current sample. *LIMDEP* is only equipped to analyze contiguous time series data.)

Observations in a cross section may be drawn randomly for purposes of bootstrapping or related analyses by using the

DRAW ; N = number of observations $

to draw without replacement, or

```
DRAW          ; N = number of observations ; Replacement $
```

to sample with replacement.

Finally, since samples often include missing observations, a switch is provided for you to instruct the program to bypass observations which include missing values during estimation. The command is

```
SKIP
```

This command should generally be used when your data have missing values. Though it is likely to be rare that you would not want to bypass these, *LIMDEP* usually does not do this automatically. The exception is in the panel data estimators, which generally do manage missing values internally.

R27.6 Multiple Imputation

Multiple imputation involves an imputation step and an estimation step. A third step, aggregation of the imputation results, takes place simultaneously with the estimation step. The imputation step involves fitting imputation equations with

```
IMPUTE        ; Lhs = the variable to be imputed
              ; Rhs = independent variables in the equation
              ; Type  = the type of variable being imputed.
```

The estimation step takes place within a procedure, as in

```
PROCEDURE $
          … commands for manipulating data, matrices, sample, etc.
          Model command ; … ; Imputation = label $
          … may be repeated
ENDPROCEDURE $
EXECUTE       ; N = number of imputations
              ; Imputation = the list of labels that appear in the procedure $
```

A trace of the imputation is requested with **; Report** in the **EXECUTE** command. Details on multiple imputation are given in Chapter R20.

R27.7 Econometric Model Estimation

The definition of a ‘model’ in *LIMDEP* consists of the modeling framework, the statement of the variables in the model, and what role the variables will play in that model. All model specifications, once again, broadly defined, will be of the same form

```
Model name    ;  variable specification, such as the name of a dependent variable
              ;  possibly other variable specifications
              ;  other information needed to complete the model specification $
```

The 'Model name' designates the modeling framework. In most cases, this defines a broad class of models, such as **POISSON** which indicates that the command is for one of the twenty or so different models for count data, most of which are extensions of the basic Poisson regression model. Many model commands provide only the class of models, and further specification is needed to provide the specific model. The form is

; Model = spec

that is used by many model commands to specify a particular variant. An example is **LOGLINEAR ; Model = Weibull… $** The following define large classes of models that are available for nearly all specific model specifications in *LIMDEP*:

; RPM	random parameters model used throughout *LIMDEP* (note, **; RPL** is a random parameters counterpart – random parameters logit model – that is used only in *NLOGIT* Version 6.) This specification may include '= list' for an extension of the model to include measured heterogeneity.
; LCM	latent class model.

The 'model variables specification' generally defines the dependent and independent variables in a model. In almost all cases, the model will include one or more dependent variables, denoted a Lhs, or 'left hand side' variable in *LIMDEP*'s command structure. Independent variables usually appear on the Rhs, or right hand side, of a model specification. The various specifications that attend the command are used to specify the basic model and to add certain optional features or model variations. Some of these are extremely general. For example, nearly every model command will contain a **; Lhs = variable(s)** specification to identify the dependent variable(s). In contrast, **; Cost** is used only by the frontier model command to request a cost (as opposed to a production) frontier model. The following lists most of the model specifications used with the model commands, in decreasing level of generality. A few specifications which have only one narrow single use in the context of only one model are omitted, and presented with the specific model.

R27.7.1 Variable Specifications in Model Commands

These essential parts of model commands are described in Chapter R8.

; Lhs = names	specifies model dependent variable(s).
; Rhs = names	specifies model independent variable(s).
; Rh1 = names	first list of variables in a two equation model.
; Rh2 = names	second list of variables in a two equation model.
; Inst = names	list of instrumental variables.
; Wts = name	weighting variable, **[,Noscale]** prevents scaling to sum to sample size.
; Hfn = names	list of variables for variance in heteroscedasticity model.
; Eqn = names	use with **SURE**/**3SLS**/**NLSURE**, multivariate probit to specify the variables or equations in a multivariate model.
; Skip= names	list of variables that should be inspected for missing values to be skipped in the current sample.
; Dfr = values	automatic creation of partially differenced values. Use with **REGRESS**.

NOTE: The variable *one* is a program created variable that always equals 1.0. Use *one* to indicate a constant term in a model.

R27.7.2 Controlling Output from Model Commands

These optional features are described in the following sections: **; Par** in Section R15.2**; Partial Effects** in Section R9.4.3, **; OLS** in Section R9.2.1, and **; Table = name** in Section R9.6.

TIMER	command – reports computation time for estimated models.
; Par	keeps ancillary parameters such as a correlation in main results vector *b*.
; Partial Effects	displays marginal effects (same as **; Marginal Effects**).
; OLS	displays least squares starting values when (and if) they are computed.
; Table = name	saves model results to be combined later in output tables.
; Matrix	reports embedded covariate matrix objects with outputs.
; Quiet	does not report model output.
; Clevel	set significance level for confidence intervals in output
; Export	exports results to .csv file for spreadsheet programs (Section R9.7.2.)

R27.7.3 Robust Asymptotic Covariance Matrices

See Section R9.4.1 for discussion of **; Covariance Matrix**. The clustering computation for robust covariance matrices is described in Section R10.2. Choice based sampling is described at several points; a reasonably detailed discussion appears in Section E27.10. Robust estimation also appears in the discussion of several models. General discussion appears in Sections R10.1 and R10.2.

; Covariance Matrix	displays estimated asymptotic covariance matrix (normally not shown), same as **; Printvc**.
; Choice	uses choice based sampling (sandwich with weighting) estimated matrix.
; Cluster = spec	computes cluster form of corrected covariance estimator. There are several extensions of this estimator.
; Robust	sandwich estimator or robust VC for **TSCS** and some discrete choice models.
; Stratum = spec	second, higher level of grouping for robust covariance with complex data sets.
; Huber	correction for cluster estimator of robust covariance matrix.
; Fpc = spec	finite population correction in cluster estimator.
; HC1	heteroscedasticity consistent covariance matrix type used by **REGRESS**.
; HC2	same.
; HC3	same.

R27.7.4 Optimization Controls for Nonlinear Optimization

These optional features are described in detail in Chapter R26.

; Start = list	gives starting values for a nonlinear model.
; Tlg[= value]	sets convergence value for gradient.
; Tlf[= value]	sets convergence value for function.
; Tlb[= value]	sets convergence value for parameters.
; Tln = value	sets convergence value for nonlinear least squares.
; Alg = name	algorithm.
; Maxit = n	maximum iterations.
; Mxit = n	restricts number of tries in line search (internal use only).
; Output = n	technical output during iterations.
; Lpt = n	Laguerre quadrature, number of points to use.
; Hpt = n	Hermite quadrature, number of points to use.
; Set	keeps current setting of optimization parameters as permanent.
; Nowarn	no warnings reported in technical iteration output.

R27.7.5 Setup for Simulation Based Estimators

; Halton	use with RPM and *NLOGIT* 6 RPL model for Halton sequences.
; Antithetical	uses antithetical pairs of random draws in simulations.
; Pts = n	number of replications to use in simulations.

R27.7.6 Execution of Procedures for Model Estimation

EXECUTE
; optional specifications $
; Bootstrap = name bootstrap estimation of covariance matrix for estimator.
; Jackknife = name jackknife estimation of covariance matrix for estimator.
; Silent estimates without reporting results.

R27.7.7 Predictions and Residuals

Fitted values (predictions) and residuals are described in Chapter R12.

; List	displays a list of fitted values with the model estimates.
; Keep = name	keeps fitted values as a new (or replacement) variable in data set.
; Res = name	keeps residuals as a new (or replacement) variable.
; Prob = name	saves probabilities as a new (or replacement) variable.
; Fill	fills missing values (outside estimating sample) for fitted values.

R27.7.8 Model Setup for Certain Models

Two step estimation is described in Chapter R18 and in numerous examples in the *Econometric Modeling Guide* (on censored data). The Harvey model is used in several model frameworks, and is described in the specific chapter in the *Econometric Modeling Guide*. The **; Model = type** specification modifies the command to request a particular form within the model class specified.

Specific forms of this specification appear in the respective chapters in the *Econometric Modeling Guide*.

; Model = name	specifies a particular form of a general model class, **SURVIVAL, LOGLINEAR, POISSON, TSCS**.
; 2step = name	two step estimation used by **PROBIT, REGRESS, TOBIT, LOGIT, POISSON**.
; Het	Harvey style model in **TOBIT, PROBIT, LOGIT, ORDERED, POISSON, SURVIVAL**.
; Hfn = list	list of variables for heteroscedastic function.
; Hfu = list	same, **FRONTIER**.
; Hfv = list	same, **FRONTIER**.
; Hf1 = list	same, **BIVARIATE PROBIT**.
; Hf2 = list	same, **BIVARIATE PROBIT**.
; Hfe = list	same, random effects linear models.

R27.7.9 Setup for Panel Data Models

LIMDEP contains an extremely large menu of panel data estimators. The set of controls listed below is used primarily with the nonlinear estimators for panel data. The data arrangement is described in Chapters R5 and in R22. Chapter R5 is used for 'one way' panels, in which the model has only a group specific effect. Models may also have a two way structure, in which there is a time specific effect. Time effects are described in Section R23.2.2. The controls listed below are discussed in numerous sections below, and summarized with the estimators in Chapters R23-R25.

Data Specification for Panel Data

; Pds = spec	specifies panel data, fixed number of periods or number given by variable.
; Time = spec	specifies time for two way fixed effects model for panel.
; Periods = t	time specification for panel estimators.
; Str = name	specifies a stratification variable for **DSTAT, REGRESS, SURVIVAL**.

Panel Data Specifications in Nonlinear Modeling Frameworks

; FEM	fixed effects model.
; Fixed	fixed effects model – used in a few cases to avoid ambiguity.
; REM	random effects model.
; Random	random effects model – used in a few cases to avoid ambiguity.
; AR1	use with *NLOGIT* 6 RPL model and all RPM autocorrelation models.
; Cor	use with *NLOGIT* 6 RPL model and all RPM for correlated random parameters.
; Cprob = name	saves conditional probabilities for panel models and *NLOGIT*.
; Group = name	use with latent class models, keeps predicted group.
; Fcn = setup	use in setup for RPM panel model and in *NLOGIT* 6 RPL model.
; Dpd	dynamic panel data models.

R27.8 Post Estimation

Post estimation analysis includes hypothesis testing, estimation and analysis of partial effects, model simulation, and decompositions.

R27.8.1 Hypothesis Tests and Restrictions

These features are described in Chapter R13.

; CML: spec	constrained maximum likelihood.
; CLS: spec	constrained least squares.
; Test: spec	Wald test of linear restrictions.
; Rst = list	specifies equality and fixed value restrictions.
; Maxit = 0	use with **; Start = list**, sets up a Lagrange multiplier test.
; Wald: spec	Wald test of linear restrictions – same as **; Test: spec**.

R27.8.2 Partial Effects

After a model is estimated,

PARTIAL EFFECTS ; Effects: variable $

Produces the average partial effect for the variable named. Additional scenarios allow analysis of the behavior of the partial effects. Use any or all of

```
; Effects: variable  |variable    = discrete set of values
                     & variable = lower (step) upper
                     @ variable = values for sample partitioning
```

Plots of partial effects are requested with

; Plot or **; Plot(ci)** for confidence limits.

R27.8.3 Oaxaca Decompositions

The model is fit to a split sample with

Model ; For [variable = *, value, value] ; … the rest of the model $

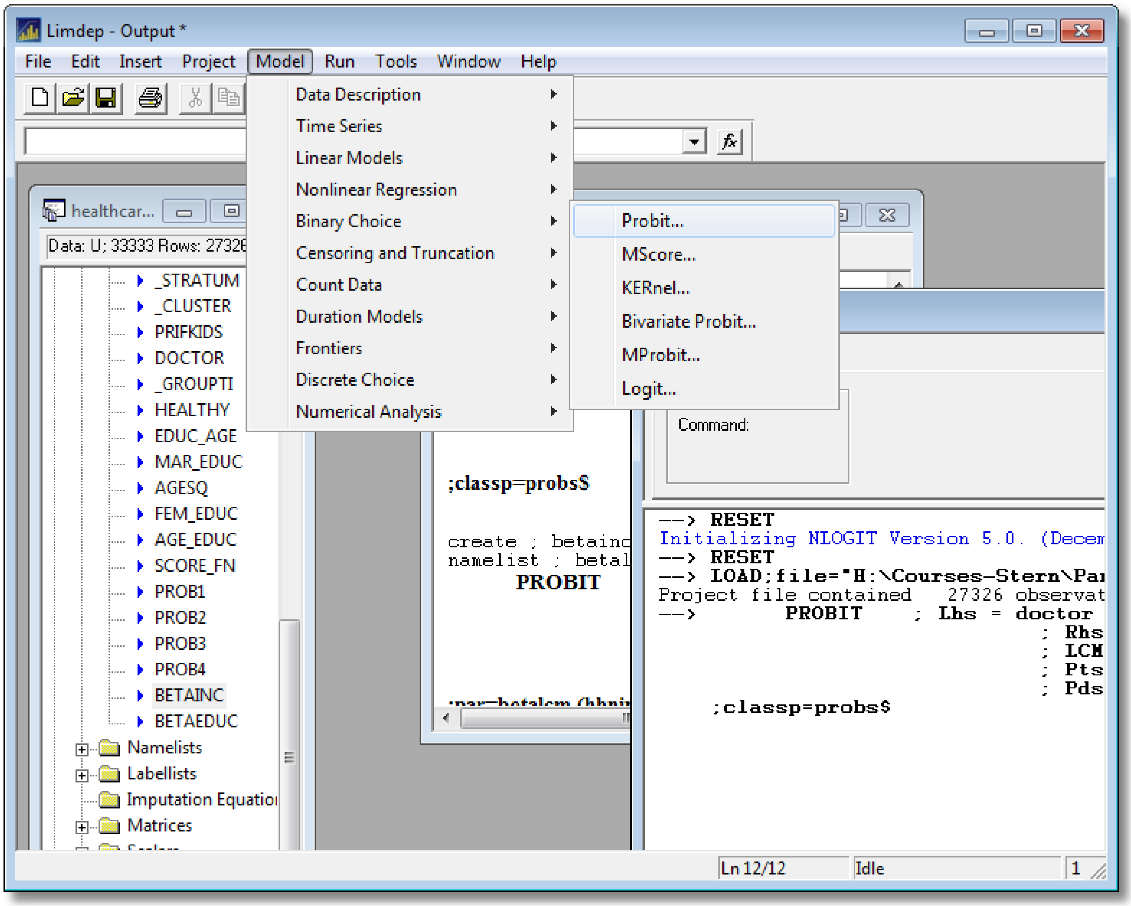
Then,

DECOMPOSE $

There are no options for this command.

R27.9 The Command Builders

An alternative method of submitting commands is to use the interactive dialog boxes which for reasons that will be evident shortly, we call the command builders. (This feature is described in detail in Chapter R8.) Command builders for model commands are produced by selecting Model in the main menu. The Model menu, shown in Figure R27.1, offers a number of groups of model frameworks. You may then select one of the groupings of models shown, to open a subsidiary menu of specific models. An example for the binary choice models is shown in Figure R27.1. You may then click a model name to open the command builder dialog box for that specific command. An example for the **PROBIT** command is shown in Figure R27.2.



```
;classp=probs$

create ; betainc
namelist ; betal
     PROBIT

;par=betalcm (hbni
```

```
--> RESET
Initializing NLOGIT Version 5.0. (Decem
--> RESET
--> LOAD;file="H:\Courses-Stern\Pa
Project file contained    27326 observat
-->           PROBIT     ; Lhs = doctor
                         ; Rhs
                         ; LCM
                         ; Pts
                         ; Pds
     ;classp=probs$
```

Figure R27.1 Selecting the Command Builder from the Model Menu

The Main tab (page) in the command builder dialog box requests the variables part of the commands. A few of the optional features will usually appear here as well, including, for example, a weighting variable. Other optional specifications are provided on the other tabs (pages) of the command builder window. As can be seen in Figure R27.2, the probit model command builder has two additional pages. *Note, you must provide the essential variable parts of a command on the Main page before you may enter the Options page. The command builder will insist on this.*

NOTE: The query (?) button at the lower left of the command builder dialog box is a link to a context sensitive Help file that contains a large amount of information about the command.

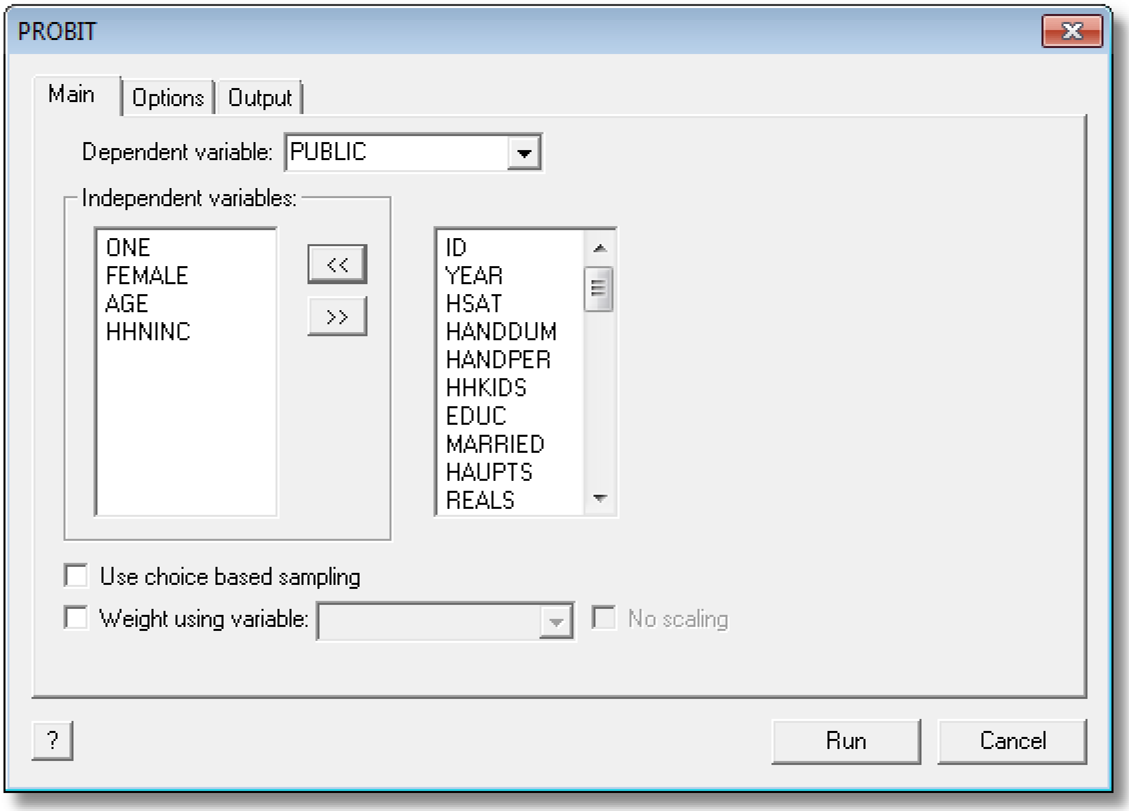
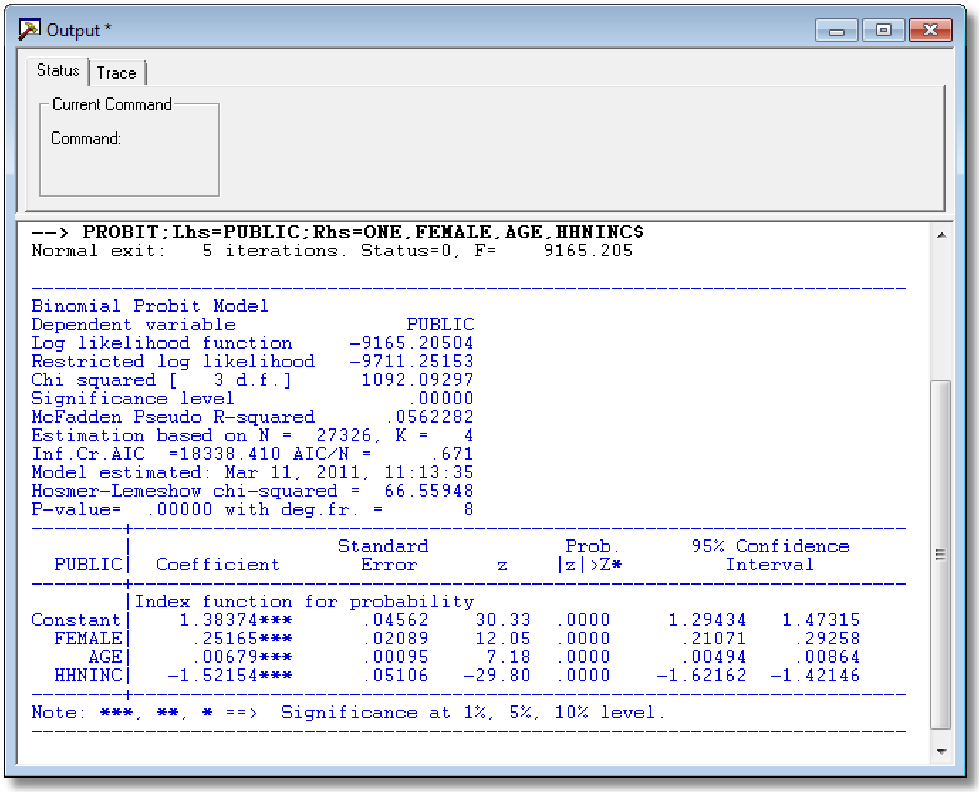


Figure R27.2 Main Page for Command Builder (PROBIT)

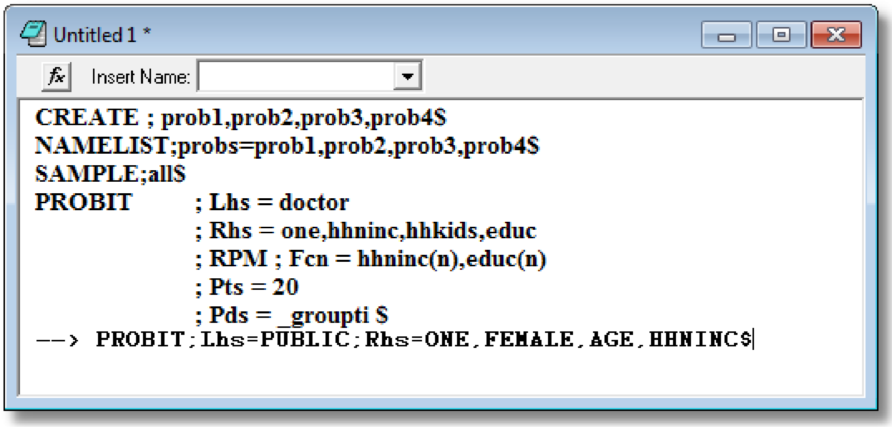
Once you have selected the model specification in the command builder window, click the Run button to submit the command to *LIMDEP* for processing. This produces two results: First, the command is carried out, and the results appear in the output window, as would result in general when a model command is issued. Second, as its name implies, the command builder 'builds' the model command, and places a copy of it in the output window with the results. (See Figure R27.3.)

The first line of text above the output is the command generated by this selection in the window. You can copy these commands from the output window and paste them into the editing window, as we have done in our example in Figure R27.4. You might find this useful if you wish to modify the model and reuse the command. The editor will usually be more convenient. Note, as well, that the command interpreter will ignore the leading '-->' so there is no absolute need to edit these characters out of the editing window.



```
--> PROBIT;Lhs=PUBLIC;Rhs=ONE,FEMALE,AGE,HHNINC$
Normal exit:   5 iterations. Status=0, F=     9165.205

-----------------------------------------------------------------------------
Binomial Probit Model
Dependent variable               PUBLIC
Log likelihood function     -9165.20504
Restricted log likelihood   -9711.25153
Chi squared [   3 d.f.]      1092.09297
Significance level               .00000
McFadden Pseudo R-squared      .0562282
Estimation based on N =  27326, K =   4
Inf.Cr.AIC  =18338.410 AIC/N =     .671
Model estimated: Mar 11, 2011, 11:13:35
Hosmer-Lemeshow chi-squared =  66.55948
P-value=  .00000 with deg.fr. =        8
--------+--------------------------------------------------------------------
        |                  Standard            Prob.       95% Confidence
  PUBLIC|  Coefficient       Error       z    |z|>Z*         Interval
--------+--------------------------------------------------------------------
        |Index function for probability
Constant|     1.38374***      .04562    30.33  .0000     1.29434   1.47315
  FEMALE|      .25165***      .02089    12.05  .0000      .21071    .29258
     AGE|      .00679***      .00095     7.18  .0000      .00494    .00864
  HHNINC|    -1.52154***      .05106   -29.80  .0000    -1.62162  -1.42146
--------+--------------------------------------------------------------------
Note: ***, **, * ==>  Significance at 1%, 5%, 10% level.
-----------------------------------------------------------------------------
```

Figure R27.3 Output Window



```
CREATE ; probl,prob2,prob3,prob4$
NAMELIST;probs=probl,prob2,prob3,prob4$
SAMPLE;all$
PROBIT       ; Lhs = doctor
             ; Rhs = one,hhninc,hhkids,educ
             ; RPM ; Fcn = hhninc(n),educ(n)
             ; Pts = 20
             ; Pds = _groupti $
--> PROBIT;Lhs=PUBLIC;Rhs=ONE,FEMALE,AGE,HHNINC$
```

Figure R27.4 Detail from the Editing Window

NOTE: The command builders are not complete. Some options and model forms must be specified with commands formed in the text editors. The command builders are intended generally for development of the more basic forms of the models and for relatively uncomplicated models. Not all optional features in all models are present in the command builder. Moreover, a few of the model frameworks are not contained in the command builder menu. We anticipate that the command builders will be used by those who are becoming accustomed to using *LIMDEP*. After a relatively short introductory period, you will probably find the text editor more convenient than the command builders.

R27.10 Econometric Data Structures and Modeling Tools

The *Econometric Modeling Guide* describes the econometric modeling frameworks that are specifically built into *LIMDEP*. The following overviews these frameworks and describes the essential commands that relate to them. Since each of these provides many options and variants, only the essential features and basic command structures are listed. Before listing the modeling frameworks, we note in this section the data structures that *LIMDEP* is designed to analyze.

R27.10.1 Cross Section Data

Most of the models and techniques that *LIMDEP* contains are best suited for cross section data. The distinguishing feature of such a data set is independence of the observations. The data will consist of a group of 'exchangeable' data points – that is, the order of the observations in the sample has no significance. Thus, regression, nonlinear modeling, optimization, etc., are typically based on sums of independent observations.

R27.10.2 Panel Data

LIMDEP contains the widest array of estimators for panel data sets available in any major package. (A summary of the panel data models in *LIMDEP* appears in Chapters R22-R25.) The panel of data consists of n groups of T_i observations. With only the exception of the TSCS model framework, whose structure makes a balanced panel necessary, panels in *LIMDEP* may always be unbalanced – no estimator requires that group sizes be equal. The range of panel data models supported by *LIMDEP* includes models for discrete choice, censored and truncated data, count data, limited range dependent variables, survival models and various models for multinomial and ordered discrete outcomes.

The panel data models supported by the program can be described mathematically as follows: The 'model' is defined by a probability model for the observed outcome,

$P(y_{it})$ = $g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it})$ where:

$P(.)$ = the probability density function of the observed random variable, y_{it}.

i = 1,...,N denotes the ith group or individual. The number of groups is sometimes unlimited, but in many cases is limited. When it is, the upper limit is 50,000.

t = 1,...,T_i denotes the tth period, ranging from one to a person or group specific T_i. With only one exception that is dictated by the structure of the TSCS model *LIMDEP always* allows T_i to vary across groups. That is, panels may always be unbalanced.

y_{it} = the observed dependent variable.

$\mathbf{x}_{it}$ = is used to denote an observed vector of independent variables. This may include variables which vary across both groups and periods, and, in some applications, may also involve variables which vary across groups but are constant across periods, such as group specific dummy variables.

$\boldsymbol{\beta}_i$	=	the parameter vector for the *i*th individual. This may vary completely across individuals, as in the random coefficients models, or it may have a fixed component and a subvector which varies across groups, as in the usual fixed effects model. It may also be constant across groups and periods, as in the random effects model.
ε_{it}	=	the stochastic component of the model. The symbol is used generically to indicate the stochastic nature of the model, not necessarily a 'disturbance.'
g	=	the density of the observed random variable conditioned on the arguments.

LIMDEP supports the following general model forms for panel data:

Fixed Effects: $g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it}) = g(\boldsymbol{\beta}'\mathbf{x}_{it} + \alpha_i, \varepsilon_{it})$

These models contain dummy variables for specific groups. Techniques that are unique to *LIMDEP* allow tens of thousands of dummy variable coefficients to be estimated in models that were previously assumed to be intractable for this approach. Roughly 50 different models, nearly all of them nonlinear, include a fixed effects form.

Random Effects: $g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it}) = g(\boldsymbol{\beta}'\mathbf{x}_{it}\ , \varepsilon_{it} + u_i)$

The econometric interpretation of this variant treats the 'effect' as an additive or multiplicative random, group specific disturbance in a model. (Many statistical treatments broaden the term to mean what we label 'random parameters' in the next paragraph.)

Random Parameters: $g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it}), f(\boldsymbol{\beta}_i)$ is defined as part of the model.

Most of *LIMDEP*'s models can be estimated in a random parameters format. Broadly, this approach bridges the Bayesian approach to estimation and the classical fixed parameters approach. Further details on this model class appear below.

Latent Class Models: $g(\boldsymbol{\beta}_i, \mathbf{x}_{it}, \varepsilon_{it}) = \text{E}_{classes}\, [\text{g}((\boldsymbol{\beta}_{class}{}'\mathbf{x}_{it}\ , \varepsilon_{it}) \mid class]$

In a latent class formulation, the continuous distribution of the heterogeneity is approximated by using a finite number of 'points of support.' The distribution is approximated by estimating the location of the support points and the mass (probability) in each interval. In implementation, it is convenient and useful to interpret this discrete approximation as producing a sorting of individuals (by heterogeneity) into J classes, $j = 1,...,J$.

Table R27.1 lists the panel data estimators contained in this version of *LIMDEP*.

Model Class	Fixed Effects	Random Effects	Random Parameters [c]	Latent Class
Linear Regression [a,d]	●	●	●	●
MIMIC[b]		●	●	
Binary Choice				
Probit [a]	●	●	●	●
Logit [a]	●	●	●	●
Complementary log log[a]	●	●	●	●
Gompertz [a]	●	●	●	●
Bivariate Probit [b]		●	●	
Biv. Probit Selection [b]	●	●	●	
Partial Observability [b]		●	●	
Multinomial Choice				
Multinomial Logit [e]		●	●	●
Multinomial Probit [b]		●		
Ordered Probability/All [a]	●	●	●	●
Count Data				
Poisson Regression [a]	●	●	●	●
Negative Binomial [a]	●	●	●	●
Poisson/NegBin ZIP [b]	●	●	●	●
Loglinear Models				
Exponential [b]	●	●	●	●
Gamma [b]	●	●	●	●
Weibull [b]	●	●	●	●
Inverse Gaussian [b]	●	●	●	●
Power [b]	●	●	●	●
Binomial [b]	●	●	●	●
Exponential regression [b]	●	●	●	●
Geometric [b]	●	●	●	●
Limited Dependent Variable				
Tobit [a]	●	●	●	●
Truncated Regression [b]	●	●	●	●
Grouped Data [b]	●	●	●	●
Sample Selection [b]	●	●	●	
Survival and Frontier Models				
Weibull [b]	●	●	●	●
Exponential [b]	●	●	●	●
Loglogistic[b]	●	●	●	●
Lognormal [b]	●	●	●	●
Stochastic Frontier [a]	●	●	●	●

Table R27.1 Model Formulations with Panel Data Estimators

[a] The random effects model can be estimated by standard REM techniques (GLS, quadrature) or by the simulation method with a random parameters formulation;
[b] The random effects model can only be estimated by the simulated random parameters approach.
[c] Any random parameters model produces a random effects model by a random constant term.
[d] Linear Regression: Fixed effects fit by maximum likelihood and least squares, random effects fit by GLS and maximum simulated likelihood.
[e] Multinomial logit with random effects is fit as a random parameters logit model by *NLOGIT* Version 6.

R27.10.3 Fixed Effects Models

The fixed effects model is

$$z_{it} = \alpha_i d_{it} + \boldsymbol{\beta}'\mathbf{x}_{it},\ i = 1,...,N,\ t = 1,...,T_i,$$

$$p(y_{it}) = g(z_{it}, \boldsymbol{\theta})$$

where α_i is the coefficient on a binary variable, d_{it}, which indicates membership in the ith group. The panel is assumed to consist of N groups with T_i observations in the ith group. The panel need not be balanced; T_i may vary across groups. Nonlinear models of this form are estimated in two ways. The *conditional* estimator is obtained by using the conditional joint distribution, $f(y_{i1},y_{i2},...,y_{iT}|\Sigma_t y_{it})$. See, for example Griliches, Hall, and Hausman (1984) who develop this for a Poisson regression. The resulting density is a function of $\boldsymbol{\beta}$ alone, which is then estimated by (conditional) maximum likelihood. This estimator is available for the binary logit, Poisson, and negative binomial models. Chapters R22 and R23 provide extensive details. Most models do not reduce to a useable closed form through this conditioning, so that the conditional estimator is unavailable. The *unconditional* estimator is obtained by a direct maximization of the full log likelihood function and estimating all parameters including the group specific constants.

The Incidental Parameters Problem

Full estimation of the fixed effects model in this fashion generally encounters the 'incidental parameters' problem. The estimators of the fixed effects coefficients are inconsistent in a fixed effects model, not because they estimate the wrong parameters, but because the variances of the estimators of α_i are of order $1/T_i$ with T_i not assumed to be increasing, not $1/N$, where N is. Thus, the properties of the slope estimator (and the estimator of θ in the negative binomial model) depend on an inconsistent estimator. The mean of the slope estimator converges to a function that deviates from $\boldsymbol{\beta}$ as a function of the extent to which the estimator of α_i deviates from α_i. Let a_i be the MLE of α_i and $\mathbf{b}$ be the estimator of $\boldsymbol{\beta}$. The usual results for the MLE in a multiparameter situation would produce consistency from the fact that $\mathbf{b} = \mathbf{b}(a_1, a_2, ...)$ and

$$\text{plim}_{N\to\infty}\,\mathbf{b} - \boldsymbol{\beta} = \text{a function of, among other results, } \text{plim}_{N\to\infty}\,a_i - \alpha_i,\ i = 1,...,N.$$

In the usual case, all terms (including the 'other results') would converge to zero. In this case, that does not hold, though the extent to which the small sample (T_i) affects $\mathbf{b}$ is unknown. Certainly if your panel contains very small group sizes, say T_i less than five or so, then this estimator is shaky. If you have fairly large group sizes, say on the order of 30 or more, then you are in the range of sample sizes that analysts often rely upon to assert other asymptotic results. Users are urged to consider this issue when using the unconditional fixed estimators.

Surprisingly, the incidental parameters problem is not present in the Poisson model. The reason for this intriguing result is that in the Poisson model, the first order conditions for estimation of the slopes are actually free of the fixed effects – see Winkelmann (2000) for a proof. This effect is illustrated in the application in Chapter E44. The conditional and unconditional estimators are identical. This is not the case for the negative binomial or binary logit models, however. It is for a few other estimators, such as the exponential regression.

LIMDEP's unconditional estimator can also produce a two way fixed effects model,

$$z_{it} = \alpha_i + \delta_t + \boldsymbol{\beta}'\mathbf{x}_{it}.$$

There will now be MaxT_i-1 additional coefficients in the model. You can request this estimator by adding

; Time = Ti

where the variable *Ti* indicates, for each observation the period in which the observation occurred. This variable must take the values 1,2,...,MaxT_i. That is, it must be coded with '*t*,' the index number of the period. A date will not work – it will be flagged as identifying too many coefficients. Do note that observations may be made at different periods in the different groups. For example, if you have a panel with three observations in the first group and seven in the second, the first three observations could have been made at t = 2, t = 4, and t = 7. The routine assumes that MaxT_i is equal to the largest group size in the model. (That way, it is assured that there are no holes in the sequence of observations.) Thus, the largest group in the sample must have this variable coded with the complete set of integers, 1,2,...,*Tmax*.

NOTE: If you have a balanced panel with **; Pds = T** where *T* is a fixed value, then you can specify the time effects with **; Time = one** as there can be no variation in the coding of the period in a balanced panel.

NOTE: Our experience has been that the time effects extension produces considerable instability in the negative binomial, though it works nicely in the Poisson model.

The fixed effects model with time effects is estimated by actually creating the time specific dummy variables. You will see a complete set of time effects in the output. As such, however, if you have a large group size in your panel, this extension may create an extremely large model.

The unconditional log likelihood is maximized by using Newton's method. A full discussion of the method is given in Chapter R23.

R27.10.4 Random Effects and Multilevel Random Effects Models

The fixed effects model is

$$z_{it} = \sigma_u u_i + \boldsymbol{\beta}'\mathbf{x}_{it},\ i = 1,...,N,\ t = 1,...,T_i,$$

$$p(y_{it}) = g(z_{it}, \boldsymbol{\theta})$$

where the common effect u_i has a distribution with mean zero and variance one – the scale of the random variable is accommodated by the unknown parameter σ_u. The distinction between fixed effects (FE) and random effects (RE) models is that while in the fixed effects case, d_{it} may be correlated with $\mathbf{x}_{it}$, in the random effects case, u_i is not correlated with $\mathbf{x}_{it}$. (Ultimately, what this means is that u_i is uncorrelated with the group mean $\overline{\mathbf{x}}_i$.) As before, all panels may be balanced or unbalanced. Estimation of the random effects model is done in three ways in *LIMDEP*.

Two Step FGLS

For the linear regression case, the simple RE may be fit by two step feasible generalized least squares. The variance components, σ_u^2 and σ_ε^2 are consistently estimated by using the ordinary least squares residuals, then the full model is fit once again by generalized least squares. This is discussed in Section E18.2.

Maximum Likelihood

With a normality assumption, the log likelihood can be formulated and maximized directly. For the linear model, this is done directly, operating on the log likelihood function itself which has a closed form. This is also the case for the nested random effects linear model discussed below. Estimation of the model is discussed in Section E18.3. For several other models, including probit, logit, tobit and ordered probit, the log likelihood function is an integral which does not have a closed form, but which can be satisfactorily approximated using Hermite quadrature.

Maximum Simulated Likelihood

Any model that can be fit as a random parameters model – Table R27.1 lists, with all variants, over 40 of them – can be fit as a random effects model by allowing only the constant term to be random in the model. This implies that virtually any model that can be fit with *LIMDEP* can be fit as a random effects model.

Multilevel Random Effects Model

Suppose the data are constructed in levels – we allow up to 10 nested levels, so that the structure is

$$v_{i,t,s,r},\ i = 1,...,N;\ \ t = 1,...,T_i;\ s = 1,...,S_{it},\ r = 1,...,R_{its}$$

where we use four levels as an example. Consider, for example, student test performance data in which *i* is school district, *t* is school, *s* is teacher, and *r* is student. Schematically, the data might appear as:

```
                i=1                     ...              i=N
|------------------------------------|...|-----------------------------|
   t=1        t=2             t=T1            t=1              t=TN
|---------| |-------| ... |---------|   |--------------|...|---------|
  s=1   s=S11 s=1 s=S12 ... s=1  s=S1,T1     s=1     s=SN1  s=1 s=2 s=SN,TN
|----| |--| |-| |---|      |-| |-----|   |----------| |-|...|-| |-| |-|
          r = 1,...,R1,T(1),S(T1)               r = 1,...,RN,T(N),S(TN)
....................................    ..............................
```

With the effects model now defined as

$$z_{itsr} = \sigma_u u_i + \sigma_v v_{it} + \sigma_w w_{its} + \sigma_\varepsilon \varepsilon_{itsr} + ...$$

we have a multilevel effects model with a potentially extremely involved correlation structure. Again, *LIMDEP* allows this up to 10 levels. In addition, the effects may be specified multiplicatively as well as additively. This model is fit by maximum simulated likelihood as a random parameters model. Random parameters models are discussed throughout the *Econometric Modeling Guide* and in some detail in Chapter R24.

R27.10.5 The Random Parameters Model

A number of researchers have analyzed a few variants of this model under different names. In particular, there is now an extensive literature on ‘mixed logit’ models (Train (1999), et al.). This multinomial logit version of this model has also been used in well known papers by Goldberg (1995) and Berry et al. (1995). Researchers, primarily not in economics, have also analyzed ‘multilevel’ models and ‘hierarchical’ models applied to linear regression, binary logit and Poisson and negative binomial regression models. These are variants of the random parameters model described here. To our knowledge, this is the first implementation of this technique in a class of models as diverse as the one listed in the preceding table.

The structure of the random parameters model from the point of view of the modeler is

$\boldsymbol{\alpha}_i$ = $[\boldsymbol{\beta}_{1i}', \boldsymbol{\beta}_{2i}', \boldsymbol{\theta}']'$

where $\boldsymbol{\theta}$ = ancillary parameters, such as the dispersion parameter in the negative binomial model – most of the models listed have no ancillary parameters

$\boldsymbol{\beta}_{1i}$ = $\boldsymbol{\beta}_1 = K_1$ nonrandom parameters, $\mathbf{x}_{1it}$ = variables multiplied by $\boldsymbol{\beta}_{1i}$

$\boldsymbol{\beta}_{2it}$ = $\boldsymbol{\beta}_2 + \boldsymbol{\Delta}\mathbf{z}_i + \boldsymbol{\Gamma}\mathbf{v}_{it} = K_2$random parameters, $\mathbf{x}_{2it}$ = variables multiplied by $\boldsymbol{\beta}_{2it}$

where $\boldsymbol{\beta}_2$ = the fixed means of the distributions for the random parameters

$\mathbf{z}_i$ = a set of M observed variables which do not vary over time and which enter the means

$\boldsymbol{\Delta}$ = coefficient matrix, $K_2{\times}M$, which forms the observation specific term in the mean

$\mathbf{v}_{it}$ = unobservable $K_2{\times}1$ latent random term in the ith observation in $\boldsymbol{\beta}_{2i}$. Each element of $\mathbf{v}_{it}$ has mean zero and variance one. Each element of $\mathbf{v}_{it}$ may be distributed as normal, uniform, or triangular. They need not be the same.

$\boldsymbol{\Gamma}$ = lower triangular or diagonal matrix which produces the covariance matrix of the random parameters, $\boldsymbol{\Omega} = \boldsymbol{\Gamma\Gamma}'$in the random effects form and $\boldsymbol{\Omega} = \boldsymbol{\Gamma}(\mathbf{I}\text{-}\mathbf{R}^2)^{-1}\boldsymbol{\Gamma}'$ in the AR(1) model.

$\boldsymbol{\beta}_i$ = $[\boldsymbol{\beta}_1', \boldsymbol{\beta}_{2it}']'$

$\mathbf{x}_{it}$ = $[\mathbf{x}_{1it}', \mathbf{x}_{2it}']'$

a_{it} = $\boldsymbol{\beta}_{it}'\mathbf{x}_{it}$

$P(y_i|\mathbf{x}_{it}, \mathbf{z}_i, \mathbf{v}_{it}) = \text{g}(y_{it}, a_{it}, \boldsymbol{\theta})$ = the density for the observed response.

Two models are used for $\mathbf{v}_{it}$:

Random Effects: $\mathbf{v}_{it} = \mathbf{v}_i$ for all t. This is the usual random effects form.

Autocorrelated [AR(1)]: $\mathbf{v}_{it} = \mathbf{R}\mathbf{v}_{i,t\text{-}1} + \mathbf{u}_{it}$ where $\mathbf{R}$ is a diagonal matrix of coefficient specific autocorrelation coefficients, and $\mathbf{u}_{it}$ satisfies the earlier specification for $\mathbf{v}_{it}$.

The multilevel random effects model described in the preceding section is also incorporated in this model by building the effects model into the random constant term of the random parameters model described here.

R27.10.6 Observations About GLIM and GEE Estimation

A fairly prominent development in the statistical literature, generalized equation estimation (GEE) modeling appears to be yet another form of estimator. (See Liang and Zeger (1986) and Diggle, Liang and Zeger, (1994).) The GEE estimator is not explicitly supported in *LIMDEP* directly as a preprogrammed routine. However, most of the internally consistent forms of GEE models (there are quite a few that are not consistent) are contained in the list in Table R27.1, so you can do several forms of GEE modeling with *LIMDEP*. As this is a frequently asked question, we consider it in detail.

GLIM

The GEE method of modeling panel data is an extension of Nelder and Wedderburn's (1972) and McCullagh and Nelder's (1983) Generalized Linear Models (GLIM) approach to specification. The generalized linear model is specified by a 'link' to the conditional mean function,

$$f(\mathrm{E}[y_{it}\,|\,\mathbf{x}_{it}]) = \boldsymbol{\beta}'\mathbf{x}_{it},$$

and a 'family' of distributions,

$$y_{it}\,|\,\mathbf{x}_{it} \sim \mathrm{g}(\boldsymbol{\beta}'\mathbf{x}_{it}, \boldsymbol{\theta})$$

where $\boldsymbol{\beta}$ and $\mathbf{x}_{it}$ are as already defined and $\boldsymbol{\theta}$ is zero or more ancillary parameters, such as the dispersion parameter in the negative binomial model (which is a GLIM). Many of the models already discussed in this manual fit into this framework, such as the standard probit model which has link function $f(.) = \Phi^{-1}(P)$ and Bernoulli distribution family and the classical normal linear regression which has link function equal to the identity function and normal distribution family. More generally, for the single index binary choice models estimated by *LIMDEP*, if $\mathrm{Prob}[y_{it} = 1] = F(\boldsymbol{\beta}'\mathbf{x}_{it})$, then this is the conditional mean function, and the link function is simply (by definition)

$$f(\mathrm{E}[y_{it}\,|\,\mathbf{x}_{it}]) = F^{-1}[F(\boldsymbol{\beta}'\mathbf{x}_{it})] = \boldsymbol{\beta}'\mathbf{x}_{it}.$$

and the distribution family is, again, the Bernoulli distribution. This form captures the parametric binary choice models, including probit, logit, Gompertz, complementary log log and Burr (scobit). A like result holds for the count models, Poisson and negative binomial, for which the link is simply the log function. So far, nothing has been added to models that are already familiar. The aforementioned authors demonstrate a method by which models which fit in this class can be estimated by a kind of iterated weighted least squares. This is one of the reasons that GLIM modeling has attracted such interest. In the absence of a preprogrammed routine, it is easy to do.

One can create a vast array of models by crossing a menu of link functions with a second menu of distributional families. Consider, for example, the matrix in Table R27.2 (which does not exhaust all the possibilities). We choose four distributional families to provide models for the four most common kinds of random variables:

		Link Functions				
Kind of r.v.	**Family**	Identity	Logit	Probit	Log	Reciprocal
Binary	Bernoulli	X	●	●	X	X
Continuous	normal	●	●	●	●	●
Count	Poisson	X	X	X	●	X
Nonnegative	gamma	X	X	X	●	X

Table R27.2 Generalized Linear Models

Nelder et al.'s estimation theory is complete in that there is no theoretical restriction on the mesh between link and family. But, in fact, most of the combinations are internally inconsistent. For example, for the binary dependent variable, only the probit and logit links make sense; the others imply a conditional mean that is not bounded by zero and one. For the continuous random variable, any link could be chosen; this just defines a linear or nonlinear regression model. For the count variable, only the log transformation insures an appropriate nonnegative mean. The logit and probit transformations imply a positive mean, but one would not want to formulate a model for counts that forces the conditional mean function to be a probability between zero and one, so these make no sense either. The same considerations rule out all but the log transformation for the gamma family. The preceding lists many of the commonly used link functions (some not listed are just alternative continuous distributions). More than half of our table is null. Of the nine combinations that are consistent, five are just nonlinear regressions, which is a much broader class than this, and one would unduly restrict themselves if they limited themselves to the GLIM framework for nonlinear regression analysis. After eliminating internally inconsistent combinations of link functions and families, nearly all of the commonly used, internally consistent generalized linear models appear as preprogrammed estimators in *LIMDEP*, though they are calibrated using maximum likelihood rather than iteratively reweighted least squares. The upshot of all this is that *LIMDEP does* fit 'generalized linear models.' GLIM is an alternative (albeit, fairly efficient) method of estimating some models that are quite routinely handled with conventional maximum likelihood estimation.

GEE Modeling

All the preceding said, GLIM has not cost anything either. GLIM provides a clever interpretation of some familiar models and an efficient algorithm. But, GEE provides a potentially useful variation of these already familiar models by extending them to panel data settings. To the preceding GLIM interpretation, the GEE approach adds what is essentially a random effects form to the panel of observations. Let us redefine the link function as

$$f(\mathrm{E}[y_{it}|\, \mathbf{x}_{it}]) \;=\; \boldsymbol{\beta}'\mathbf{x}_{it} \;+\; \varepsilon_{it},\ t = 1,...,T_i.$$

Now, consider some different approaches to formulating the $T_i \times T_i$ covariance matrix for the heterogeneity: (Once again, we borrow some nomenclature from the GEE literature):

Independent:	$\mathrm{Corr}[\varepsilon_{it}, \varepsilon_{is}] \;=\; 0,\ t \neq s$
Exchangeable:	$\mathrm{Corr}[\varepsilon_{it}, \varepsilon_{is}] \;=\; \rho,\ t \neq s$
AR(1):	$\mathrm{Corr}[\varepsilon_{it}, \varepsilon_{is}] \;=\; \rho^{\lvert t-s \rvert},\ t \neq s$
Nonstationary:	$\mathrm{Corr}[\varepsilon_{it}, \varepsilon_{is}] \;=\; \rho_{ts},\ t \neq s,\ \lvert t-s \rvert \leq g$
Unstructured:	$\mathrm{Corr}[\varepsilon_{it}, \varepsilon_{is}] \;=\; \rho_{ts},\ t \neq s.$

The GEE approach to estimation is a form of generalized method of moments. Most of these models are already available in other forms. The first one is obvious - this is just the pooled estimator ignoring any group effects. The second is the random effects model. We have noted a large number of models, including most of those in the valid set of GLIMs that *LIMDEP* can fit in the random effects form. In addition, all models that are available in the random parameters form can be fit with just a random constant term and can thus provide this random effects model. This includes most of the GLIM models and some others, such as the tobit model. In addition, the random parameters model allows an AR(1) format for the random constant term, so all the models that fit in the exchangeable case can also be fit as in the AR(1) case. *LIMDEP* has no facility for the nonstationary or unstructured cases. We do note, however, these sorts of models are very weakly identified in any estimation setting, owing to the large number of parameters that must be estimated to characterize the distribution of an unobserved random vector. A fully unstructured correlation matrix, for example, is nearly inestimable as an ancillary parameter in a model fit by maximum likelihood, because the log likelihood becomes quite flat in the space of the correlations. If the panel is at all large, users should not be optimistic about fitting models such as the unstructured one above. (For example, *LIMDEP*'s multinomial probit and multivariate probit models face this difficulty.)

In this respect, then, *LIMDEP* can estimate most GEE models. The estimation technique however, is simulated maximum likelihood, not the method of moments. By construction, *LIMDEP*'s estimator will be more efficient asymptotically, though in the sizes typical of panel data sets, this will probably be a minor consideration.

We note, finally, although there are a few GEE models not available in *LIMDEP*, the ability to structure the random parameters model with random coefficients on all variables makes this estimator, in fact, far more general than the GEE estimator. The end result would be, in answer to the frequently asked question, yes, *LIMDEP* does do GLIM and GEE estimation, and considerably more with the random parameters model.

R27.10.7 Latent Class Models

A model for a panel of data, $i = 1,...,N$, $t = 1,...,T_i$ is specified

$$P[y_{it}|\,\mathbf{x}_{it}] = F(y_{it},\boldsymbol{\beta}'\mathbf{x}_{it}) = P(i,t).$$

Henceforth, we use the term ‘group’ to indicate the T_i observations on respondent i in periods $t = 1,...,T_i$. (In our formulation of this model framework, T_i may equal one – this all applies to cross sections as well as panels.) Unobserved heterogeneity in the distribution of y_{it} is assumed to impact the density in the form of a random effect. The continuous distribution of the heterogeneity is approximated by using a finite number of ‘points of support.’ The distribution is approximated by estimating the location of the support points and the mass (probability) in each interval. In implementation, it is convenient and useful to interpret this discrete approximation as producing a sorting of individuals (by heterogeneity) into J classes, $j = 1,...,J$. (Since this is an approximation, J is chosen by the analyst.)

Thus, we modify the model for a latent sorting of y_{it} into J ‘classes’ with a model which allows for heterogeneity as follows: The probability of observing y_{it} given that regime j applies is

$$P(i,t|j) = P[y_{it}|\,\mathbf{x}_{it}, j]$$

where the density is now specific to the group. The analyst does not observe directly which class, $j = 1,...,J$ generated observation $y_{it}|j$, and class membership must be estimated. Heckman and Singer (1984) suggest a simple form of the class variation in which only the constant term varies across the classes. This would produce the model

$$P(i,t|j) = F[y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \delta_j], \text{Prob}[\text{class} = j] = F_j$$

We formulate this approximation more generally as,

$$P(i,t|j) = F[y_{it}, \boldsymbol{\beta}'\mathbf{x}_{it} + \boldsymbol{\delta}_j'\mathbf{x}_{it}], F_j = \exp(\theta_j) / \Sigma_j\exp(\theta_j), \text{ with } \theta_J = 0.$$

In this formulation, each group has its own parameter vector, $\boldsymbol{\beta}_j' = \boldsymbol{\beta} + \boldsymbol{\delta}_j$, though the variables that enter the mean are assumed to be the same. (This can be changed by imposing restrictions on the full parameter vector, as described below.) This allows the Heckman and Singer formulation as a special case by imposing restrictions on the parameters. (A further generalization is discussed below.)

The latent heterogeneity model can be extended by allowing measured influences in the prior probability. Let z_{i1}, ..., z_{im} denote M time invariant variables (such as sex, marital status, location, education) which affect the latent class probabilities. Then, we extend the model so that prior class assignment is formulated as a multinomial logit;

$$P[\text{class } j \mid \mathbf{z}_i] = F_{ij} = \frac{\exp(\boldsymbol{\theta}_j'\mathbf{z}_i)}{\sum_{j=1}^{J}\exp(\boldsymbol{\theta}_j'\mathbf{z}_i)}$$

R27.10.8 Time Series Data

LIMDEP contains some capabilities for time series data, including Box-Jenkins identification, ARMAX and distributed lag models, GARCH models of several sorts, tests for unit roots, and a few techniques for estimating dynamic equations.

R27.11 Econometric Model Estimation Templates

LIMDEP's preprogrammed estimation routines include a very wide variety of models and variants of model forms. We have arranged the documentation of these estimators in the *Econometric Modeling Guide* by the class of estimation method or the modeling frameworks of interest – in the form of an econometric reference – rather than by program command. The next chapter will lay out the essential format of the model estimation commands in *LIMDEP*. Only the command structures will be presented. Mathematical details on specific models and techniques are given in the indicated places in the *Econometric Modeling Guide*.

R28: Diagnostics and Error Messages

R28.1 Introduction

The following is a complete list of diagnostics that will be issued by *LIMDEP* and *NLOGIT*. Altogether, there are well over 1,000 specific conditions that are picked up by the command translation and computation programs. Nearly all of the error messages listed below identify problems in commands that you have provided for the command translator to parse and then to pass on to the computation programs.

Most diagnostics are self explanatory and will be obvious. For example,

```
82  ;LHS - variable in list is not in the variable names table.
```

states that your Lhs variable in a model command does not exist. No doubt this is due to a typographical error – the name is misspelled. Other diagnostics are more complicated, and in many cases, it is not quite possible to be precise about the error. Thus, in many cases, a diagnostic will say something like ‘the following string contains an unidentified name’ and a part of your command will be listed – the implication is that the error is somewhere in the listed string. Finally, some diagnostics are based on information that is specific to a variable or an observation at the point at which it occurs. In that case, the diagnostic may identify a particular observation or value. In the listing below, we use the conventions:

<AAAAAAAA>	indicates a variable name that will appear in the diagnostic,
<nnnnnnnnnnn>	indicates an integer value, often an observation number, that is given,
<xxxxxxxxxxxx>	indicates a specific value that may be invalid, such as a ‘time’ that is negative.

The listing below contains the diagnostics and, in some cases, additional points that may help you to find and/or fix the problem. The actual diagnostic you will see in your output window is shown in the Courier font, such as appears in diagnostic 82 above.

We note it should be extremely rare, but occasionally, an error message will occur for reasons that are not really related to the computation in progress. (We cannot give an example – if we knew where it was, we would remove the source before it occurred.) You will always know exactly what command produces a diagnostic – an echo of that command will appear directly above the error message in the output window. So, if an absolutely unfathomable error message shows up, try simplifying the command that precedes it to its bare essentials, and by building it up, reveal the source of the problem.

Finally, there are the ‘program crashes.’ Obviously, we hope that these never occur, but they do. The usual ones are division by zero and exponent overflow. Once again, we cannot give specific warnings about these, since if we could, we would fix the problem. If you do get one of these and you cannot get around it, please contact us at support@limdep.com.

R28.2 Optimization

The following messages occur during estimation of a model. In some cases, estimation must stop at that point – the iterative process has broken down. In a few cases, the error is actually just a warning or notification of some temporary condition, in which case, estimation will continue. In this case, you will want to look closely at the final results and the accompanying output to see if any further problems have come up during optimization. Note that some of these warnings occur without a diagnostic number. These are denoted 'wrn' in the listing below.

NOTE: Section R26.4 contains lengthy discussion of most of these diagnostics and their causes.

801 `Problem with starting values provided.`
Exit status will be 5. Check the command. This means the starting values could not be read from the command.

802 `Cannot compute function at start values.`
Exit status will be 4. This is not likely with internal values; it usually happens if you give a bad set of starting values.

`Cannot compute function at current values.`
Exit status will be 4, breakdown of the iterations. For example, if an estimated variance becomes negative and it is not possible to retreat to a valid value, iteration is halted with this error.

803
```
Hessian is not positive definite at start values.
B0 is too far from solution for Newton method.
Switching to BFGS as a better solution method.
```
You can usually ignore this. Unless the starting values are very good, this is common. BFGS does not need the Hessian. But, this may be a warning of bad things to come.

804
```
Looks like convergence occurred too quickly.
NOTE: Convergence in initial iterations is rarely at a true
function optimum.  Check all results.
```
Most problems take more than four or five iterations. When one does not, it can mean that the starting values are already the solutions (OK), or the derivatives are zero at the starting values (possibly not OK). This is application dependent. It may not be a problem.

```
Looks like convergence occurred too quickly.
Note: DFP and BFGS usually take more than 4 or 5 iterations
to converge.  If this problem was not structured for quick
convergence, you might want to examine results closely. If
convergence is too early, tighten convergence with, e.g.,
;TLG=1.D-9.
```

805 `Initial iterations cannot improve function.`
Exit status will be 3. This may be due to unusable starting values, or the starting values are already the solution.

```
Wrn    Hessian is not definite at current values.
       Switching to BFGS (gradient based) method.
       (Not a failure. Just looking for a better algorithm.)
```
This is the same as 803, but it occurs after iterations have begun.

```
806    Line search does not improve fn. Exit iterations.
```
Exit status will be 3. This is usually not an error. If the likelihood function is fairly flat near the maximum, this will occur. However, for a very badly behaved log likelihood, this error will occur, and will give you a warning about the function being optimized. If it happens in the first iteration, the model is probably inappropriate for this data set.

```
Wrn    Maximum iterations reached.
```
Exit iterations with status=1.

```
Wrn    Abnormal exit from iterations.
```
If current results are shown, check the convergence values shown in the results. The results shown may not be a solution (especially if the initial iterations stopped). This is a general failure. Some other diagnostic will indicate the cause.

```
Wrn    Smallest abs. parameter change from start value = <xxxxxx>.
       Note:  At least one parameter did not leave start value.
```
This is a general failure. If you programmed derivatives in MAXIMIZE, at least one of them is always zero. More generally, if starting values are so bad, it may not be possible to make progress toward a solution.

```
Wrn    Iterations aborted by user request.
```
Exit status will be minus 1. You clicked the Stop button.

R28.3 Setup and Runtime Diagnostics

Most of the diagnostics listed here are produced by errors in commands, not by problems that arise during estimation of a model. Most of these diagnostics are self explanatory. A few suggestions will be added where that may not be the case. Note, some diagnostic numbers have multiple diagnostics, which will be grouped and listed with that number. In general, diagnostic numbers are only meaningful to help you navigate through this listing. The numbers attached to diagnostics are used internally in the program, but not specifically meaningful in this listing.

```
  1    Unrecognized command.  (Missing ; ?).
```
The command is not one of the recognized commands. This can happen if you have a $ in the middle of a command and you submit extra lines. For example, the command(s)

```
PROBIT      ; Lhs = y ; Rhs = x $
            ; Het
```

will cause this error if you submit both lines. The $ in the first makes the second look like a new command.

2 `Command is more than 10,000 characters.`
This is usually caused by a command file that contains its own data. It should rarely happen. The program is protected internally from this. However, you might actually be trying to execute such a huge command. If so, it needs to be modified.

Diagnostics 3 through 14 are produced when you try to read a data file with IMPORT or READ.

3 `READ - ; not  NREC,NVAR,FORM,NAME,FILE,STAR,BYVA.`
The READ command has an unrecognized code following a semicolon.

4 `READ - error reading NREC (NOBS) or NREC was not given.`
This should only occur if NOBS specifies a value that cannot be read or is not positive.

5 `READ - error reading NVAR or NVAR not specified.`

6 `READ: Syntax error in format.  Missing paren?  Other error?`

7 `READ - Data set is too large. Expand data area (Project...).`

8 `READ - too many variables.`
This is not likely. The limit is 899.

9 `READ - error or end of file occurs while reading data set.`
Unexpected end of file will usually not be the problem. A file with bad data in it, such as alphabetic data in a Fortran formatted file can cause this.

10 `READ -error or end of file reading variable names or format.`

11 `READ -Converting ;BLANKS. Command must give ;FORMAT=(...).`

12 `READ -Names=list or in file. Not enough names were given.`
Check this against NVAR given in the command.

13 `Expected OPEN;INPUT=..., OPEN;OUPUT=..., LOAD/SAVE;FILE=...`

14 `READ - unformatted read, a record contains erroneous data.`

15 `OPEN - expected = sign not found.`
The command is supposed to be OPEN ; Input = the name of the file $.

16 `OPEN - expected ; or $ was not found.`
The command must end with a $. This is an unlikely error, as a $ or ; is going to be found eventually.

17 `OPEN - expected to find INPUT=filename or OUTPUT=filename.`

18 `READ;...;BLANKS. Maximum line length = 500. Line is too long.`

19 `OPEN, LOAD, SAVE, READ, or WRITE. Could not open the file.`
This may mean the pointer to the file did not actually locate the file. Try enclosing a file name in double quotes. This is necessary if the file name has spaces in it. It may also occur if some other program is using the file.

20 `ROWS - The value is too small.  Just use SAMPLE;1-value $.`

21 `CALC - Too many subexpressions in parens. Unable to compile.`

22 `READ. An invalid name spec was found in file or command.`
Names must begin with a letter and have no more than eight characters.

23 `Warning: F or chi-squared <= 0. Check regression for errors.`

24 `File system error.  Cannot OPEN the indicated file.`
The disk drive may be empty, or the file may be in use by other software.

25 `MATRIX;{calc command}expression. Did not find closing },),].`
This error often occurs when something else is wrong. For example, if your MATRIX command contains a variable name where a matrix name is expected, or there is a comma out of place, this error can occur.

26 `REJECT or TVC-Wrong number of operators >, &,... Mismatched.`

27 `REJECT or TVC - &, +, or $ was found where >,<,... expected.`

28 `REJECT or TVC - >,<,... was found where &, +, or $ expected.`

29 `REJECT:0 obs. in resulting sample. Current sample restored.`

30 `CREATE/CALC ;...Unmatched parentheses in current subcommand.`

31 `Panel model. You have only one group. Check STR/PER variable.`

32 `Model command.  Start values. Unreadable or wrong # given.`

33 `PROC,EXEC,OPEN,SAVE,LOAD,SHOW in proc.`
These are commands that cannot appear in procedures.

34 `Proc. buffer full, 10,000 characters or > 50 commands in a proc.`

35 `REJECT/SAMPLE  -  Maximum sample size for command exceeded.`

36 `SAMPLE - did not find ; or $ where expected.`

37 `SAMPLE - value given in sample specification not a  number.`

38 `SAMPLE - range n1-n2, n1<=0,n2<=0, n1>nrec, n2>nrec, n1>n2.`
A SAMPLE command must have a range of values from low to high. The maximum sample size may be seen at the top of the project window – this is nrec.

39 `Poisson.  Invalid limit given for censoring model.`

40 `Poisson.  Truncation model requested, but no limit given.`
; Truncation must be accompanied by ; Limit = value.

41 `Error in variable list given for HFN, HF1, or HF2.`
Check the list of variables. At least one of them does not exist.

42 `MATRIX - in IF[value1 rr value2] - ] precedes value2.`
This is a syntax error. It looks like the closing bracket is in the wrong place.

43 `ORDERED PROBIT - Current estimates thresholds  not ordered.`
This is only a warning that occurs during estimation. Results will follow. It can usually be ignored, but if the procedure breaks down, this is likely to have preceded the failure. If you have a large sample and your dependent variable almost never takes at least one of the interior values, expect this diagnostic. Note diagnostic 44.

44 `ORDE,Panel,BIVA PROBIT: A cell has (almost) no observations.`
Estimation of the thresholds requires values in all cells. A (nearly) empty cell makes estimation of the thresholds impossible.

45 `LOGNORMAL REGRESSION - nonpositive values for lhs variable.`

46 `REGR;CLS: - Specified constraints not linearly independent.`

47 `GROUPED DATA - must have at only 3-18 cells (limit values).`

48 `NAMELIST name conflict. Could not construct namelist.`
This will be accompanied by an explanation, such as the name you tried to use was already in use for a matrix or scalar.

49 `GROUPED Y > # limits + 1. Bad data or too few limit values.`

50 `Complex roots: Cannot compute eigenvalues of matrix.`
It was not possible to obtain a solution. Some matrices produce this. It is not an error.

51 `CREATE - did not find expected = in expression.`

52 `CREATE/DATA or Probit;HOLD. No room for a new variable.`

53 `CREATE/REJECT. Expression too complex. Over 50 parentheses.`

54 `NTOBIT: You must specify two LHS variables for this model.`

55 `HFN, HF1, or HF2. Too many variables. The limit is 75.`

56 `Not used.`

57 `List in DTA function contains an unknown variable.`
This diagnostic is no longer used. It should not occur. If it does, use CREATE ; Namelist name = matrix name $ instead.

58 `CREATE-Data must be alone w/o IF() or other transformation.`
When you move a matrix into a variable or namelist, do not include any sort of condition, such as If(...).

59 `Cannot ID the name above.  (Not a Matrix,Variable,Function).`
The transformation in the CREATE command contains an unrecognized name. Some transformations allow any of the types above, so all the tables are searched before this error occurs.

60 `CREATE - Arguments in a DOT function are not conformable.`

61 `Compilation error in CREATE. See previous diagnostic.`

62 `SURE using MLE - Model is too big. # xs  > 100 or # ys  > 20.`

63 `SURE w/MLE - Missing or bad Pattern list or bad CLS: list.`

64 `CREATE - in parsing IF( ... )... did not find closing ).`

65 `MATRIX;name=value*result. Problem interpreting value.`

66 `REGR ; PANEL ....  This command must provide ;STR=variable.`

67 `Matrix element in an expression has an invalid subscript.`

68 `MATRIX;name=result. Name is a reserved MATRIX name (e.g. B).`

69 `WALD,NLSQ,MAXIMIZE. Cannot translate the listed string.`

70 `Cannot compute TSCS with only one group.`

71 `Variable list contains a name not in the expected table.`
This is a general error that can occur in many contexts. The command involves a list of variables, and at least one of them does not exist. Check the names against the project list.

72 `All models. List of names, including namelist is too long.`

73 `Expected , or ; or $ in list of names was not found.`

74 `NAMELIST - Table is full; can only keep 10 NAMELISTs.`

75 `NAMELIST: Syntax error. Expected = or DELETE not found.`

76 `EXECUTE - no procedure has been stored yet.`

77 `NAMELIST - name in a namelist is not a valid variable name.`

78 `NAMELIST - Namelist may contain up to 100 variable names.`

79 `EXECUTE - specification  not ;N=n or ;T(j)=value or ;QUERY.`

80 `EXECUTE;N=value$ value is invalid or unreadable as a number.`

81 `Model command - specification aaa in ;aaa not recognized.`

82 `;LHS - variable in list is not in the variable names table.`

83 `RHS/RH1 variable or MATRIX in list not in the names table.`

84 `INST/RH2/SKIP  variable in list  not in the var. name table.`

85 `;WTS - variable not in table or nonpositive weight found.`

86 `Not used.`

87 `EXECUTE;T(j)=c$ j is not 1, 2, or 3 or c is erroneous.`

88 `QR or LDV model.  Found a bad value for dependent variable.`

89 `REGR;ALG=GRID(l,u,d) - one of the values is not readable.`

90 `;MAXIT=n - n is not a valid number.`

91 `;OUTPUT=n - n is not a valid number.`

92 `REGR;PLOT(variable) - variable is not in the names table.`

93 `REGR;ALG=GRID(l,u,d) - invalid value given for l, u, or d.`

94 `Model command must include the ;RHS or ;RH1 specification.`

95 `;SEP=variable name - variable name not in the names table.`

96 `;LIMITS=item,item,.. - item is not a valid name or number.`

97 `The model command must include the ;LHS specification.`

98 `The model command must include ;RH2 or ;INST specification.`

99 `;PDS/INT/RHO/PTS/PERIODS=n. Bad # or inconsistent if panel.`
In the ; Pds = value form, in a panel, the value must be reasonable, and the full sample size must be an even multiple of it.

100 `;HOLD(*) - * not NDX, or IMR; no , or ); or unknown name.`

101 `LOGIT - one of the cells (outcomes) has no observations.`

102 `LOGIT - likelihood cannot be computed at current estimates.`

103 `LOGIT - Number of LHS vars is too large. Bad model setup?`

104 `DISC - Singular Hessian.`
Look for an attribute that does not vary across choices. If you have one, use ; Rh2 for it.

105 `ARMAX or other model. Unable to forecast with missing data.`

106 `DISC - A choice was (almost) never chosen, empty cell.`

107 `DISC - Observations in data not a multiple of # of choices.`

108 `DISCRETE CHOICE;CHOICES=list... - Name has > 8 characters.`

109 `;CLS/RST - error in syntax.  Check specified constraints.`

110 `;RH1=list - a variable in the list is not in names table.`

111 `;RH2=list - a variable in the list is not in names table.`

112 `;CHOICES=list - DISCRETE CHOICE requires this spec.`

113 `PROBIT;START=list;LOAD or INCI or BIVA. Check start values.`

114 `CALC - RAN(seed) - seed is not readable.`

115 `CALC ; DOT(z1,z2) must be (var,var) or (matrix,matrix).`

116 `CALC - Unable to compute result. Check earlier message.`
This is a general diagnostic that is issued after it becomes impossible to translate a command or calculation. An earlier diagnostic will show the error.

117 `CALC Vectors in DOT(x,y) have different number of elements.`

118 `CALC The name used is reserved for estimation results.`

119 `MATRIX - QFRI or QFII. Inner matrix of q-form is singular.`

120 `CALC Unable to find a solution for internal rate of return.`

121 `MATRIX - QFRI or QFII. Quadratic form is singular.`
The quadratic form can be computed, but the result cannot be inverted.

122 `CALC - Cannot identify matrix given in ROW or COL command.`

123 `MATRIX ; name(subscript) = value expected, bad syntax.`

124 `MATRIX - name=result. Name is a reserved CALC name (e.g. S).`

125 `CALC ;Rn=fcn(list) too few or bad values given for function.`

126 `CALC zero divide or function cannot be computed at value(s).`
This error also occurs when the result of a calculation will be infinitely large or small.

The following errors arise when computing a set of least squares coefficients. Many models use this program to compute starting values, so these OLS generated errors can arise in the context of almost any model.

127 `Models - Sample sum of weights is less than or equal to 1.0.`

128 `Models - Insufficient variation in dependent variable.`

129 `Models - Variable i (i is given) has no variation.`

130 `Models - Regression; insufficient degrees of freedom.`

131 `Models - Regression; regressors are collinear.`

132 `Models - Regression; sum of weights < number of parameters.`

The following errors will occur during estimation of a specific model. They will precede or occur at the same time as the 800+ errors listed in Section R28.2.

133 `MINIMIZE/NLSQ/SURE: Syntax error in or missing LABELS=list.`

134 `Models - Unusable starting values.  Unable to continue.`

135 `Models - singular Hessian during Newton iterations.`

136 `Models - Maximum iterations exceeded by Newton iterations.`

137 `Iterations: function not computable at crnt. trial estimates.`

138 `Models - Maximum iterations in Steepest Descent or DFP/BFGS.`

139 `Line search no longer improving function. Check results.`
The iterations are probably about to terminate. Check the derivatives. This may be near enough to the maximum for the results to be useable. If need be, add ; Output = 3 to the model command to produce a list of derivatives.

140 `Not used.`

141 `Iterations - current or start estimate of sigma nonpositive.`
This is a warning. It often happens during iterations, but a solution is obtained anyway. Just means a bad trial value. See Section R26.4.7 for discussion.

141 `Survival scale parameter not positive`
`Selection scale parameter not positive`
These errors occur during estimation of a survival model subject to sample selection. They are transient warnings, not errors. Estimation will continue

142 `Estimated correlation is outside the range -1 < r < 1.`
Same sort of problem as in error 141.

143 `Models - estimated variance matrix of estimates is singular.`
This often happens after what looks like successful estimation. Try reducing the model to find out why it occurs.

In the following matrix diagnostics, there are many references to functions that now have simple equivalents that will usually avoid the diagnostic. For example, PART(b,1,value) might now be b(1:value) and Xcpm(X) would normally be X'X.

144 `MATRIX - INDX - matrix has different # of columns from LHS.`

145 `MATRIX - INDX - expected namelist or variable name on LHS.`

146 `MATRIX - C(j)=CHNG(B1,B2). B1,B2 must have same dimensions.`

147 `MATRIX - C(j)=NORM(B1),CHNG(B1,B2). B1, B2 must be vectors.`

148 `MATRIX - C(j)=NORM(B1) or C(j)=CHNG(B1) - B1 not defined.`

149 `MATRIX - C(j)=proc().proc() must be NORM, CHNG, or a number.`

150 `MATRIX - expected ; or $ or }, ], or )not found.`

151 `MATRIX - expected =  not found where expected.`

152 `MATRIX - MATRIX;C(j)=...$ j is not 1, 2, or 3.`

153 `MATRIX - names table is full (100 names).  Use DELETE.`

154 `MATRIX - procedure name (name = proc(list)) not recognized.`

155 `MATRIX - Procedure requires a computed, not a data matrix.`

156 `MATRIX - Procedure requires a data, not a computed matrix.`

157 `MATRIX - A=proc(list)$ There is a bad value in list.`

158 `MATRIX - A=LOAD(list)$ list contains wrong number of values.`

159 `MATRIX - A=LOAD(list)$ Number rows or columns not positive.`

160 `MATRIX - A=PART(B...)$ syntax.  , or ; or ) not found.`

161 `MATRIX - A=PART(B,list)$ list must contain 2 or 4 values.`

162 `MATRIX - A=PART(B,r1,r2,c1,c2)$ invalid for dimensions of B.`

163 `MATRIX - A=proc(B...)$ B is not the table of matrix names.`

164 `MATRIX - A=INIT(r,c,v),IDEN(r),IPDL(r,c). Bad or short list.`

165 `MATRIX - A=INIT(),IDEN(),IPDL(), row or column invalid.  0?`

166 MATRIX - A=INDX(B,...), # of columns of A and B must match.

167 MATRIX - matrix name in proc(list) is not in table.

168 MATRIX - namelist in output of proc may not contain ONE.

169 MATRIX - A=[B1/B2]. Matrices must have same number of columns.

170 MATRIX - A=[B1,B2]. Matrices must have same number of rows.

171 MATRIX - GINV, SINV, SQRT, ISQR, DTRM - nonsquare matrix.

172 MATRIX - VECD(B) or RNDM(,sigma) - matrix B is not square.

173 MATRIX - DIAG(B) - matrix B is not a row or column vector.

174 MATRIX - CVEC(B) - matrix B is not square.

175 MATRIX - procedure requires two matrices.

176 MATRIX - QFRM/QROW-B1 and B2 not conformable for QFRM/QROW.

177 MATRIX - MSUM or MDIF, matrices must have same dimensions.

178 MATRIX:MPRD(B1,B2,...) or MPLOT, matrices not conformable.

179 MATRIX - MTPR or MTTP(B1,B2) matrices are not conformable.

180 MATRIX - RNDM(mu,sigma). Mismatch of dimensions of mu & sigma

181 MATRIX - MIPR(B1,B2) - B1*B2 is not square - cannot invert.

182 MATRIX - MPRI(B1,B2) - B1tB2 is not square; cannot invert.

183 MATRIX - MIPR(B1,B2) - B1 and B2 are not conformable.

184 MATRIX - SCLR or REPL - lhs matrix must already exist.

185 MATRIX - GINV,SINV,CHOL singular, not P.D. if SINV or CHOL.

186 MATRIX - SQRT, ISQR, or, ORTH - nonpositive root.

187 MATRIX - MIPR(B1,B2) - B1 is singular; unable to proceed.

188 MATRIX - MPRI(B1,B2) - B1tB2 is singular.

189 MATRIX - XDOT,XCPM,XVCM,XCOR,XLSQ - namelist must be first.

190 MATRIX - XDOT,XCPM,XVCM,XCOR,XLSQ:namelist or var. is 2nd.

191 MATRIX - XDOT,XCPM,XVCM,XCOR,XLSQ;variable must be 3rd.

192 MATRIX - XLSQ - columns of specified matrix are collinear.

```
193   MATRIX - result is too large for buffer. (rows*cols> 22500).

194   MATRIX - XMLT,XSTD,XORN,PCOM - LHS namelist does not exist.

195   MATRIX - XMLT - expected matrix name or variable on RHS.

196   MATRIX - XMLT - RHS matrix not square, same # columns as X.

197   MATRIX - XSTD or XORN - a variable in list has no variation.

198   MATRIX - XORN - Correlation matrix has a nonpositive root.

199   MATRIX - PCOM - Invalid matrix name given on RHS.

200   2SLS - Too few instrumental variables or they are collinear.

201   SELECT - selection leaves insufficient degrees of freedom.

202   SELECT - variance matrix for criterion equation is singular.

203   POISSON or NEGBIN: Negative or noninteger value in LHS var.

204   DSTAT;PDS=N...$ - matrix for partial autocors is singular.

205   CREATE;namel=namer[-lag]$  lag is not a valid integer.

206   CREATE;namel=namer[-lag]$Syntax error, [ not where expected.

207   CREATE;namel=namer[-lag]$namer must be an existing variable.

208   MINIMIZE;...$  Start values not given or contain bad values.

209   MINIMIZE; The IV estimator is only for nonlinear 2SLS.

210   MINIMIZE;FIX:...$ Too many fixed values, or bad spec.

211   SURE or 3SLS;.. EQn variable list has an unknown variable.

212   SURE or 3SLS:EQn=list. n is invalid, nonpositive or > 20.

213   SURE or 3SLS;resid. cov. mat. singular;LHS vars. collinear?

214   Not used.

215   Model too large.K>150,>120 for SELE,>100 for 2SLS or MINI.

216   REGR, SURE, or 3SLS.  Params*# of constraints must be < 401.

217   SELECT must be preceded by PROBIT or LOGIT with ;HOLD.

218   MATRIX;STAT(B,VB)  problem with names B or VB.

219   MATRIX;STAT(B,VB)  B or VB has inappropriate dimensions.
220   SAVE/LOAD -- Read or write error with binary file.

221   The data on the LHS variable appear not to be coded 0,1,2...

222   Input file: Read error reading command.

223   Any regression: OLS gives a perfect fit; check model.

224   LIMDEP/NLOGIT: FIML not enabled. This feature not available.

225   CREATE;x=DTA(m);...$   DTA must be the only transformation.

226   MATRIX; a list of values is given but not assigned a name.

227   Matrix;[list/list/...] lengths of lists are inconsistent.

228   Matrix loaded with [list/list/...] Bad values or unknown name.

229   Matrix: Unable to translate command. Check for syntax error.

230   PROC=name{...}$ Expected to find }$ to end command. Syntax.

231   Cannot accurately compute probabilities for F(p,1,1),P>.999.

232   Noninteger degrees of freedom given for t, F, or chi-sqrd.

233   Matrix - Out of room. Unable to save result of computation.

234   Matrix - name=QFRI(A,B)... B is not positive definite.

235   NLSURE requires more than one LHS variable/Equation.

236   APPEND - there are insufficient rows to append the data.

237   APPEND - Requires too many new variables to be added.

238   WARNING!! All 3 convergence rules =0. Will never exit iters!

239   Test:... or Cls:... the j in B(j)... is > # of coefficients.

240   ;LIMITS=matrix... bad specification, too many or bad values.

241   Poisson/NegBin: Cannot compute function - extreme values.

242   NLSQ:Wrong number of start values - need one for each label.

243   File system error. Files may not be opened by PROCEDURES

244   SELECT. Corrected s.e. < 0.  Check for bad or missing data.

245   READ: NOBS too large to store.

246   Poisson/NegBin w/ panel. Cannot compute fn at current parms.
```

247 `Not used.`

248 `2 way panel data, # periods > 20000?? Check ;PERIOD variable.`

249 `Random effects. Did not find positive estimated component.`

250 `SURE/3SLS; SIGMA = matrix. Matrix was not in the table.`

251 `;STR=variable name (ORDE,SURV,CRMO/PANEL). Name not found.`
Stratification variables are used in several places. This is a general diagnostic. The error is also produced by MATRIX panel data functions, and the ; Sep = variable specification in SWITCH.

252 `Not used.`

253 `REGR;DFR=list.  List contains a bad or unreadable value.`

254 `REGR; robust OLS variance estimator. Model too big. (K>100).`

255 `MATRIX; Matrices not conformable for operation. Check sizes.`

256 `CREATE:Internal table overflow. Break command into parts.`

257 `SURV:...There is insufficient space left in the data array.`

258 `SURV Stratification variable has too many or invalid values.`

```
Too many effects: Obs=<NNNNNN> Group=<NNNNNN> Limit=<NNNNNN>
Invalid group ID for effects model. I=<NNNNNN> ID=<NNNNNN>
No obs. found for group <NNNNNN> in effects model.
```
These diagnostics occur with the Cox model with fixed effects. They will usually occur because of a badly coded group identifier or stratification indicator.

259 `SURV;PLOT... Unable to find regressor vector(s) for plotting.`

260 `SURV;PLOT Regressor vectors have wrong number of elements.`

261 `SURV Maximum of 100000 observations exceeded. Cannot continue.`

262 `SURV;STR... Invalid stratification. Unable to continue.`

263 `Not used.`

264 `;Keep=name and/or ;Res=name - no room left for new variable.`

265 `POIS;MODEL=N... No evidence of overdispersion. Use Poisson.`

266 `EXEC;rname=F,L[,D]. No repetitions.int(|L-F|[/|D|]) is zero.`

267 `EXEC;RNAME=F,L,D. One of F<L & D<0 or F>L & D>0 or D=0.`

268 `EXEC;RNAME=F,L,D.  excessive repetitions (over 30000!).`

269 EXEC;RNAME=list.No scalars left to create the loop index.

270 Not used.

271 GMM: Covariance matrix for moments or estimates is singular.

272 2SLS or SELECT. Using 2sls. Negative R-squared. (Warning.)

273 SURV ; RH2 = ... $ Inadequate censoring split for model.

274 SURV;RH2... $ Not available for Prop. Haz. Forget MODEL=..?

275 SURV;Log-linear model. Data are not in log-form. Transform.

276 Model...;LIMITS=list. Too many values. Limit is 20.

277 WKx/DIF/XLS file has an unexpected format. Cannot continue.

278 WKx/DIF/XLS.Numeric data found where var name was expected.

279 READ. Data input from a WKx file must be stored internally.

280 Box-Cox. The model cannot be fit with the values provided.
The Box-Cox transformation cannot be computed for negative values.

281 OPEN or WRITE error on last command. Disk problem? Missing?

282 LOAD command. Problem with file OPEN or READ. Restarting.

283 SURVIVAL. No variation in LHS variable! Cannot compute.

284 Output file is corrupted. Closing to keep current contents.

285 WRITE or READ requires a file name. (WRITE/READ;FILE=...)

286 WRITE command. ;FORMAT=something unidentifiable. Expected (.

287 > 899 variables in project file! StatTransfer or DBMS Copy?

288 RENAME:The NEW name is already in use. Use a different one.

289 Not used.

290 RECODE Error: Syntax, Number of values or Unreadable value.

291 Error in RENAME command. Must be OLD_NAME = VALID_NEW_NAME.

292 Error in STRING command. Must be STj = string, j=1,2, or 3.

293 MATRIX MVEC(name,r,c). name is not the name of a vector.

294 MATRIX command contains unbalanced (), <>, [], or {}.

295 `MATRIX MVEC(name,r,c). Too few elements to make rxc matrix.`

296 `MATRIX MVEC(name,r,c). Too many elements for an rxc matrix.`

297 `Invalid # rows in matrix of quad/bilinear form in MINIMIZE.`

298 `Quad/bilinear form badly dimensioned. Cannot compute it.`

299 `READ;...;NKMAX=value.  Value is <200000 or invalid.`

300 `Not used.`

301 `LOAD. File too large. Use Project:Settings/Data area to reset.`

302 `Calc. No room for new scalars. Use DELETE to make space.`

303 `CREATE. Division by a zero standard deviation.`

304 `CREATE has too many subcommands (> 100). Break up command.`

305 `CREATE: Errors occurred which prevented transformations.`

306 `CALC. NTB. P too close to 0 or 1.  Can’t get X accurately.`

307 `;TLF=f ;TLB=b ;TLG=g  Value given, f, b, or g is unreadable.`

308 `;SMOOTH=value for MSCORE. Value is not readable.`

309 `;TIES=n, ;END=n, ;QNT=q for MSCORE. Value given is bad.`

310 `Computing restricted least squares. R*VAR(b)*R’ singular.`

311 `Probit: Data on Y are badly coded. (<0,1> and <=0 or >= 1).`

312 `Ordered: Bad stratification variable or too many strata.`

313 `Stepwise. No variables pass selection rule to enter. Y = a.`

314 `Residual Plot. Too many to residuals to plot.`

315 `Stoch. Frontier: OLS residuals have wrong skew. OLS is MLE.`
This is a theoretical issue, not a program problem. If the OLS residuals are skewed in the wrong direction, the MLE for the stochastic frontier is OLS. This usually means that there is no evidence of inefficiency in your data. We emphasize (since this is a frequently asked question), this is a data issue. When this condition arises, it is because the data and the model, at least the current specification of it, are inconsistent. There is no ‘fix’ other than a different model or specification.

316 `Crosstab: One of the variables is always <= 0 or >= 50.`

317 `Plot. Number of values is too large for plotter. (> 15000.)`

318 Identify ; Rhs = variable. You forgot to include ;PDS = T.

319 Dstat;RHS=LIST ; Quantiles $ Too many observations (>22000).
This will also arise if the sample is > 4000 for plots or > 100000 for histogram.

320 ;2Step=NAME. Is NAME OK? Preceded by Probit or Logit?

321 SURV:Proportional Hazards. Fixed values setup is incorrect.

322 MSCORE. Too many observations.(10000 is the limit.)

323 Panel:Sum[N(i)] < K+1. Check STR variable. Use Matr;Gsiz(var)$.

324 READ;FILE=name...$ The data file was not found when expected.

325 MATRIX:Panel. Stratification variable is bad or missing.

326 Work space overflow. Too many lines of names in data file.

327 Closing command file to prevent reading data as commands.

328 FORMAT in READ;BLANKS. Slash format may not be inside parens.

329 FORMAT in READ;BLANKS. Parens may not be nested to 3 levels.

330 FORMAT in READ;BLANKS. Repetition N(...). Error reading N.

331 REGR;PANEL. ONE found in RHS with 2SLS. Redo without ONE.

332 READ. Your data set is too big to fit internally.

333 FORMAT provided for READ command has unbalanced parens.

334 ROWS;max$ The value given is unreadable, < 100, or too big.

335 SURV;...$ Unable to obtain start values for Gompertz model.

336 SELECT w/ 2SLS. Mismatched numbers of RHS & INST variables.

337 SELECT;bivar probit: Unable to find DELTA1 and/or DELTA2.

338 Selection data are incorrectly coded. Unable to continue.
This is produced by the two treatment model.

339 SURV ; LIMITS ...$ You have given more than 2 limit values.

340 SURV;WTS=matrix for S. The matrix you gave is the wrong size.

341 SURV with TVCs. (log) Normal model is not available here.

342 SURV with TVCs. Truncation not allowed in this model.

343 SURV with TVCs. Splitting model not allowed in this model.

344 LOGIT/Fixed effects. Cannot compute P. (Sum(yi) near 0 or T).

345 Note:MAXIT=0 set by user. LM stat. kept in scalar LMSTAT.
This is not an error. The program is telling you what it is doing with the statistic.

346 SURVIVAL. Unstable estimates. Unable to plot distribution.

347 ORDERED Probit or Logit. Too many cells. The limit is 50.

348 ORDE;HAZ;ENDPOINTS=list$ Bad vals., wrong number, not ordered.

349 TSCS. Not enough workspace for your problem. K or N too big.

350 TSCS. Inconsistent (or nonexistent) values for T. PDS=????

351 FRON:Cannot have both PDS and an NTIME variable in command.

352 Model with Panel. Sum of N(i) not equal to full sample size.

353 Internal error. Unable to convert a date. Check last command.

354 DATE;YEAR[.mth] or [.qtr]$ Year must be 4 digits. < 1000 ?

355 DATE;YEAR.??$ Expected valid month or quarter not found.

356 DATE;YEAR.Q$ found, but Q is not 1, 2, 3, or 4 as expected.

357 PERIOD command found. Your data are undated. Use DATE first.

358 PERIOD command is not of the form PERIOD ; Begin - End $.

359 PERIOD command specifies invalid date or END before BEGIN.

360 CREATE:Parameter in STD(x),XBR(x),DEV(x) must be a variable.

361 HISTOGRAM:Too many limits (> 39) or intervals (> 38) given.

362 HISTOGRAM:No values found in specified or default ranges.

363 MATRIX:XDOT or other moment matrix. Missing data. No result.

364 FPLOT: Command must include ;PTS = number of points to plot.

365 FPLOT: Command must include ;LIMITS=lower,upper for range.

366 FPLOT: ;LIMITS=L,U. L and U must bracket the start value.

367 FPLOT: Command must include ;PLOT(label).

368 PROBIT;Random Effects. K too big (>40) or T too big (> 75).

369 PROBIT;Panel:Options RH2 and grouped data not allowed.

370	`PROBIT;Random Effects: Problem is too big. K x T > 1000.`
371	`MATRIX: LADB function is limited to K>=1 & K<21 and N<=5000.`
372	`CALC: You may not change the loop variable of EXEC;loop=..$`
373	`CALC: Do not use N on the left of an equals sign.(Reserved).`
374	`MSCORE ... ; TIES $ File error opening work file for ties.`
375	`Not used.`
376	`Not used.`
377	`Command OPEN;FILE=name$ is not valid. Use INPUT or OUTPUT.`
378	`EA/LimDep - TSCS models are limited to 20 groups.`
379	`EA/LimDep - SURE/3SLS are limited to 5 equations.`
380	`A READ error has occurred while reading from your file.`
381	`An error occurred opening the file requested.` The file name is given.
382	`An end of file error has occurred reloading from the file.`
383	`LHS must supply 1 matrix, C, for simplex method.`
384	`LinProg:No. of limits in Xl or Xu must = no. of values in C.`
385	`INPUT files cannot open other input files.`
386	`LinProg:Limit specifications, ;LIMITS=Xl,Xu needed for LP.`
387	`SAVE  LOAD  SHOW must be followed by ;FILE=name$.`
388	`LinearProg:Number of activities is limited to 300 for LP.`
389	`File conflict. Opening current input for output or vice versa.`
390	`READ/WRITE;UNIT=number. number was unreadable. Check command.`
391	`LnPrg:Matrix A must contain NX+2 cols. NX=no. of values in C.`
392	`WRITE;format=binary... File name must be given for binary.`
393	`LP: Number of constraints must not exceed 300 (rows in A).`
394	`BOXCOX:Values too large. Model is explosive. Scale your data.`
395	`BOXCOX:;LAMBDA=value not found or MODEL=3 & no THETA found.`

396 `BOXCOX:With ;PTS=n, you must give lower,upper or l,u,theta.`

397 `BOXCOX:Weights for hetero. model must be > 1. Rescale W(i).`

398 `Bivar.Probit:Hessian not PD. WESML VC not computed. BHHH used.`

399 `MNAME(first:last). Check syntax. First or last is bad.`
This is produced by extracting part of a matrix.

400 `MNAME(first:last).  Matrix size exceeded by this address.`
You have specified bad dimensions. Check the matrix size.

401 `MATRIX. CXRT/complex roots. Nonconvergence. Cannot compute.`

402 `Name conflict. This will cause some commands not to work.`

403 `Using default name:YFIT for prediction/RESID for residuals.`
Either the name you gave was not useable, or no name was given.

404 `PROBIT. More than one LHS variable given. Grouped? Use ORDE?`

405 `WALD test. VC matrix specified is wrong size or not square.`

406 `Not used.`

407 `Not used.`

408 `WALT test. No functions -- FNj=... -- found on command line.`

409 `Matr;name(*,J)=vec. Not a vector, unknown matrix, or bad J.`

410 `HREG:Estimates diverging. Variances vanishing or exploding.`

411 `ARMAX: Moving average terms are explosive. Exit iterations.`
This tends to happen when you are using the wrong model.

412 `ARMAX: Model has too many parameters. Unable to estimate.`

413 `ARMAX: Model has no MA terms. Use REGRESS to estimate it.`

414 `ARMAX: Not enough observations to fit model.T-d-p-q-K < 11.`

415 `ARMAX: Singular derivatives matrix Cannot invert GtG.`

416 `ARMAX:More than 20 iterations.Cannot find initial MA terms.`
The model is not consistent with the data. This is the wrong model for these data.

417 `ARMAX: You tried to forecast beyond maximum ROWS of data.`

418 `SURE/3SLS: The AR1 models are not available for 3SLS.`

419 `Improper function setup. Expected BVN(x,y,r). Ambiguous.`

420	`DISC: Count variable for number of choices must be wrong.`
421	`MODEL;... Current sample setting has no observations.`
422	`PROC=a new name. You already have 10 procs. defined.`
423	`EXEC ; PROC = name.  PROC does not exist yet. Cannot run.`
424	`REGR;PANEL;AR1: AR1 is not available with a two way model.`
425	`REGR;PANEL. Could not invert VC matrix for Hausman test.`
426	`NLSUR;LHS=list. Number of variables not equal to # of eqns.`
427	`NLSUR or WALD. You must specify eqns with ;Fn1=...;Fn2=...`
428	`NLSUR. Unknown name given with ;SIGMA=name.`
429	`NLSUR ;SIGMA=name. Wrong dimensions. (R or C not = # eqns.)`
430	`NLSUR ;SIGMA=name. SIGMA matrix is not positive definite.`
431	`DISC;nested logit. Not enough work space for VC matrix.`
432	`Not used.`
433	`SELECT..;with 2 treatments. Did not find VDELTA (bi-probit).`
434	`Matrix: [a] or <a> defines diag. matrix > than 150x150.`
435	`You can use a different setup to avoid huge matrices.` This is just a tip, not an error. *LIMDEP* almost never needs to compute huge matrices.
436	`Matrix: <var.> creates vector longer than 22500 elements.`
437	`Matrix: <matrix> attempts to invert a nonsquare matrix.`
438	`Matrix: <matrix>. Cannot invert. The matrix is singular.`
439	`Matrix: <matrix>. No inverse. Zeros on the diagonal.`
440	`Matrix: [result].  This construction is not meaningful.`
441	`Matrix:..<..>.. or ..[..].. Did not find closing > or ].`
442	`Matrix: The expression contains an unknown name.`
443	`Matrix: ..<..>.. or ..[..].. Term must define a vector.`
444	`Matrix: <scalar>. Scalar is 0. Cannot take reciprocal.`

445 `Matrix: Diagonal middle matrix has wrong number of terms.`

446 `Matrix: Matrices are not conformable for multiplication.`

447 `Current estimated covariance matrix for slopes is singular.`

448 `More than 10 DOxx loops in PROC. Limit is 10.`

449 `DO... Syntax error. Expected DO..;label;control $.`

450 `Found NAMELIST:index. Invalid index, < 0; > # of variables.`

451 `Maximum iterations. Exit status for parameter search = 2.`
This is a generic diagnostic used by the optimization routine.

```
Counters for panel data:
Counter <=0:<nnnnnn>. Row=<nnnnn> group=<nnnnn>.
Counter variable does not match the sample.
```
These diagnostics are produced by the CREATE command for computing moments for panel data.

```
Could not compute ADF. See previous diagnostic.
Unstable equation in lagged differences. Sum is >= 1.
Perfect fit in ADF regression equation. EstVar[c]=0.
Sample has too few observations to carry out test.
Moment matrix for ADF (regression) test is singular.
```
These diagnostics are generated by CALC when attempting to compute the augmented Dickey Fuller test statistic.

```
You must OPEN;EXPORT=file before exporting results.
Write error exporting matrices to CSV file.
```
MATRIX and CALC will both check for problems exporting results to a CSV file.

```
Number of start values must match # vars. in RHS.
Did not find */-+ or > in scenario specification.
Unknown variable name in scenario specification.
Variable in scenario does not appear in the RHS.
Bad value found in scenario specification after = sign.
```
These diagnostics are produced by the BINARY CHOICE command used after PROBIT or LOGIT to analyze the predicted probabilities.

```
Using ;PDS=number. NOBS not a multiple of number.
Group count variable is mismatched to full sample.
The number of variables for TABLES must be less than 11.
Invalid weighting variable. Nonpositive values were found.
Number of strata > 5000! Tables overflowing. Must exit.
Use only one of ;PDS, ;CLUSTER or ;STR in the command.
```
These diagnostics are produced by the TABLES command for analyzing panel data on a variable.

```
Quantile Regression needs nK <= 200,000 and n < 10,000.
```

452 `Cannot sample <nnnnnn> from  <nnnnnn> without replacement.`
Your DRAW command specifies more draws than there are observations in the sample.

```
Bootstrap sample exceeds 100,000 obs. Must exit.
DRAW command for panel data.
Base sample for panel/bootstrap must be < 400,000.
DRAW ; PDS=spec. Could not identify spec.
Counter for panel group sizes is nonpositive.
Sum of counts is > full sample. Check variable.
Bootstrap sample is > 20000 groups. Must exit.
```
These diagnostics are produced by the DRAW command for panel data.

```
Cannot compute ROC for N > 375,000 points, for N = ...
```

```
Cannot sample <nnnnnn> from <nnnnnn> without replacement.
Bootstrap sample exceeds 100,000 obs. Must exit.
Maximum current sample for bootstrap is 780,000.
```
These errors occur when using bootstrapping with panel data

453 `Expected DRAW;N=number$ (;REP optional). Check syntax.`
`Invalid N for DRAW, < 0 or > 100000 (20000 for panel)`

454 `This BIVARIATE probit model needs two LHS variables.`

455 `NAME....: Obs. in sample= nnnnnn, VALUE.`
You have specified a badly coded variable for bivariate probit.

456 `Obs= nnnnn Sum of Pij= x.xxxx. Should be 1.`
Proportions data for bivariate probit do not add to 1.0

457 `Selection bivar. probit states ;LIMITS=ja,jb.`
The values ja and jb must be 1 or 0.

458 `Selection: two treatments, looking for DELTA1 and DELTA2.`
Either the matrices were not found or the ones found are the wrong size.

459 `Selection: two treatments, looking for and not finding VDELTA.`

460 `Expression/eqn is too complex. Too many subexpressions,`
`3 value functions, scalars, or operations.`
Simplify. This is from MAXIMIZE. It means your complicated expression has too many components, and an internal table has overflowed. You must break the function into parts, use subfunctions, or use more than one command.

461 `LHS variable for binary choice model is not binary!`

462 `0/1 choice model is inestimable. Bad variable = <AAAAAAAA>.`

463 `Its values predict 1[<AAAAAAAA> = 0 or 1] perfectly.`

464 `Error in or near ‘FFF...’`
This is for the CALC command. The ‘FFF...’ will be the part of the command that could not be translated.

465 `CALC function, cannot identify a name in expression.`
This depends on the context in terms of what kind of name the expression is looking for.

466 `A CALC expression has mix of matrices and other things.`
This diagnostic depends on context. Dot products, for example, must be two matrices or two variables.

467 `CALC: Dot product, vectors not the same size.`

468 `AAAAAAAA may not appear in an expression.`
This gives the name of an entity, such as a namelist, that may not appear in the expression.

469 `Current Rho = <value given>, using value to compute S2(e).`
The current value of RHO is not useable. A recent valid value is being used instead.

Diagnostics 470 - 485 are produced by computations of functions and expressions by CALC.

470 `Bad observation for CALC function <1 or too big.`
This depends on the function.

471 `Cannot compute  <function given>; Parameter <nn>, is <description>.`
The value might be too small, too large, less than one, etc. This depends on the function.

472 `Did not find a variable in this NGI function.`
This is the group size function in CALC.

```
Test of proportions requires binary variables.
TST(x,y) function requires matrices &/or variables.
A sample is too small to carry out the test of equality.
A variable given for TST function has no variance.
Rank Correlation.
The function requires one or more variables.
RKC and CNC require at least two variables.
Harmonic and Geometric Means.
Harmonic mean needs a nonzero scale.
A variable in XGM or XHM is not a set of ranks 1,...,n.
Cannot compute geo./har. mean with negative values.
```
These diagnostics are produced by setups for the Tst function in CALC, used for testing equality of means, variances or proportions.

473 `Cannot compute matrix function with this parameter.`
This depends on the function.

474 `Did not find the variable for MIN, MAX, RMX function.`

475 `Too many observations (>22000) for Med(...) function in CALC.`

476 `Not able to ID variable in SUM, XBR, SDV, VAR function in CALC.`

477 `NAMELIST: more than 1 name in SUM, XBR, SDV, VAR function in CALC.`

478 `REGRESSION function RSQ etc. has a nonvariable in it.`
This is a CALC function.

479 `CHK function needs both RHS and LHS of a model.`
This is a CALC function.

480 `CALC Computing RSQ, ESS, ... found missing values.`

481 `Binomial, Geometric, NegBin. Pi must be in [.05,.95].`
This is a CALC function.

482
```
Binomial, Hypergeometric: n must be in [2,25].
Poisson: Lambda must be > 0 and no more than 15.
Hypergeometric, NegBin., M must be in [1,99].
Hypergeometric, P must be < 100.
Hypergeometric: m must be less than P.
```
These are all CALC functions used for drawing discrete probability distributions.

`Found [ELSE], but no prior IF[...] was set.`
This is a logical condition in CALC.

483 `Replications for MVN probabilities, bad or > 2000.`

484 `NAME on LHS of CALC function is not a valid name.`

485 `Cannot identify this name: <name is given>.`
This is a matrix function in CALC.

486
```
TVC <nn> has too many operations (max=15).
Unable to identify operand <nn> in TVC <n>.
Cannot compute TVC <nn> Observation=<nnnnnn>.
```
This is for the Cox model with TVC. Check the specification of the TVC.

```
Data error. Individual <nnnnnn> has <nnnnnn> records?
Data error, Gompertz. Individual <nnnnn> record <nnn>
T=<xxxxx>.
Data error. Individual <nnnnnn> T0=xxxxx, T1=xxxxx. Invalid
values.
Data ended expecting <nn> more lines for individual <nnnnnn>.
Found <nnnnnnn>  records, and <nnnnn> individuals.
Unexpected data configuration.
Gompertz model: Observation <nnnnn> duration < 0.
```
These diagnostics result from the parametric survival models with a sequence of observation specific values per individual, denoted by <nnnnn>.

487 `<AAAAAAAA> is an invalid name. Names may not contain <character given>.`

488 `<AAAAAAAA> is an invalid name. Names may not begin with <character given>.`

489 `CONFLICT:  <type of name> name  <name is given> already used as a <type, e.g., matrix>.`

Diagnostics 490 - 495 are generated by specification of the SURE model estimated by MLE.

490 `Label  <AAAAAAAA> is repeated in labels list.`

491 `A name in the pattern is not among the labels.`

492 `Your pattern list has the wrong number of specifications in it.`

493 `Col <n> of your parameter matrix is all zeros.`

494 `Row <n> of your parameter matrix is all zeros.`

495 `Label  <AAAAAAAA> does not appear in pattern matrix.`

496 `Namelist  <AAAAAA> is no longer defined.`
This is a warning. It is the result of deleting a variable.

497 `Cannot compute function at current values.`
This occurs during optimization. See Section R26.4.7 for related diagnostics. This may stop iterations if no nearby value is known.

498 `Skipping <nn> repetitions after error flag set.`
This occurs during bootstrap iterations when invoked by EXECUTE command.

```
Setup FOR[variable(=values)] missing ]. Cannot continue.
In FOR[variable...] variable name is not recognized.
In FOR{variable=list], list is unreadable or > 100 values.
In FOR[variable=list], list item is not an integer.
FOR[variable] implies more than 100 repetitions.
```
These diagnostics are produced by problems in setting up a model command with the conditional MODEL ; For [condition] ; ... $

498 `Cannot use FOR[variable] during bootstrap iterations`
`No nonempty subsamples were found!`

499 `DSTAT, Stratification variable value is > 50.`
There are too many strata.

500 `Variable  <AAAAAAAA> has no variation. Cannot compute ACF.`

501 `Bootstrap could not find estimated parameter  <AAAAAAAA>.`
Check the EXECUTE command.

501 `GROUP=Name. Cannot identify the name. Check vars.`
`Histogram can only plot 4 groups at a time.`
`Data for multiple histograms must be 0 < x < 40`
`Data for multiple histograms must be integers 0-39`

502 `Matrix specified in bootstrap setup is not a vector.`

503 `Too few bootstraps were run to finish analysis.`
The estimator wants at least as many replications as there are parameters.

504 `Estimated result,  <name given> never changed.`
Bootstrap repetitions did not produce any variation in the item being estimated.

505 `Negbin/ZIP/Het is over specified/inestimable. Use POISSON.`
This happens if estimated θ goes to zero.

506 `Use POIS ; ZIP ; ... for LOGIT w/o heterogeneity.`
The model specification appears inconsistent. It looks like a request for a selection model, but is not specified correctly. This is a guess.

507 `Insufficient selected observations to fit Poisson with selectivity.`

508 `Matrix panel function, list of vars is not right.`
The specification must be variable(s) followed by a stratification variable. Either the wrong number of variables is given or they do not look right for the function. This is function specific.

509 `GSUM( specification ) must begin with a namelist then`
`comma....MATRIX function)`

510 `Did not recognize namelist in GSUM function.`

511 `Observation= <nnnnnn>,  Variable= <xxx.xxxx>) bad stratum.`
The value for the stratification looks inappropriate.

512 `Target name for SCL function must be < 7 characters.`
`The value given for the SCL function must not be a variable.`
`Did not find opening and closing parens for SCL fn.`
`Expected NAMELIST comma VARIABLE in SCL(.) not found`
`NAMELIST specified inside SCL function does not exist.`
`VARIABLE specified inside SCL function does not exist.`
`SCL function creates new NAMELIST. 10 already exist.`
These diagnostics are for the Scl function in CREATE.

```
      STR variable has missing values. Cannot continue.
      Stratum > 200,000 obs. Table has overflowed.
      Nonsense value for group size. Cannot continue.
      Panel moments. Cannot ID left side matrix.
      Variable or list name on RHS of moment is not recognized.
      Variable name after [ is not recognized.
      In [residual,panel # or name], cannot ID panel spec.
      Fixed panel group size given is not positive.
      Sample size is not a multiple of fixed group size.
      Sum of variable group sizes is > sample size.
      Group means fn. must provide variable and panel spec.
      Data area is full. No room to store group means var.
      Group means: Variable does not exist in data set yet.
      Cannot replace variable with own means. Use a new name.
      Group means panel spec must be ;STR=spec or ;PDS=spec
      GroupMeans(Variable,pds). Pds is not a # or variable.
```

These diagnostics are for the Group Means function in CREATE.

```
      Can only RANK one variable per CREATE command.
      Syntax error in CREATE;Name=Rnk(variable).
      Cannot locate variable in Rnk(variable) function.
      Rnk(.) can only be applied up to N = 100,000 obs.
```

These diagnostics are for the Rnk(.) function in CREATE

```
      Create;[Lag]=value$ Could not read value.

      Could not compute sample. See preceding diagnostic.
```

This occurs when attempting to use the Mvn function to create a random sample.

```
513   Cannot identify variable in EXPAND function.

514   MATRIX in CREATE; ... = matrix. Name not in table.

515   Namelist created by matrix move to data: conflict.

516   HISTOGRAM : Bad limits= <nnn> and <nnn>.
      Histogram can only plot 4 groups at a time.
      GROUP=Name. Cannot identify the name. Check vars.
      Data for multiple histograms must be integers 0-39
      You gave <nn> ;LABELS for <nn> groups.

517   Cannot plot. No variation in LHS variable.
      Cannot plot. No variation in RHS variable.
      Cannot produce centipede plot with > 1 LHS variable
      Multiple plots is only available with PLOT command
      Limits and Endpoints must be determined internally.
      Data must be sorted (internally) for multiple plots.
      ;Regression is not available with multiple plots
      Cannot stratify data with multiple plots.
      Multiple plots is limited to 5 LHS and RHS variables.
      Multiple plot needs same number of RHS and LHS vars.

518   Stratified plot must use PLOT for 1 RHS & 1 LHS variable.
```

519 `Demonstration program does not support READ or APPEND.`
`Sorry, NLOGIT_ACA does not support IMPORT or APPEND.`

520 `Cannot rearrange a matrix after READ.`
`Labels cannot be installed by APPEND.'`
`Error in ;LABELS=column. Cannot continue.`

521 `Use matrix transpose to rearrange matrix after READ.`
You tried to read your data into a matrix with the By Variables specification. Just read the data into the matrix, then transpose the matrix.

522 `Not Used.`

523 `Cannot merge files into a matrix.`
This diagnostic is produced by the merge feature in READ invoked with GROUP = variable.

524
```
Cannot merge file, no variables exist yet.
Cannot merge data arranged By Variables.
APPEND cannot be used to merge data sets.
Cannot merge data in .WK1/XLS/DIF files.
Did not find your GROUP variable:  <AAAAAAAA>.
Data set is not a multiple of group size.
Bad group counter = <nn>. Row= <nnnnnn>,  group = <nnnnnn>.
You must specify NOBS to expand GROUPED data.
Reading only  <nnnnnn> obs into <nnnnnn> groups.
```
These diagnostics are also produced by the merge feature in READ invoked with GROUP = variable. There is a mismatch between the two files.

525 `Cannot rearrange a binary file into a matrix.`

526 `Extract from binary file. You must give COLS=...`

527 `Problem with worksheet file. Nonsense dimensions.`
`Found NREC= <nnnnnnnn>,  NVAR= <nnnnnnnn>.`

527 `DIAG='ByVar,;FORMAT and ;Blanks not supported with Labels=j'`

528 `Same as 527.`

529 `RECODE must begin with NAME(...=...,...).`

530 `Invalid number given for recode. Unable to recode.`

531 `Closing parenthesis not found where expected.`
This is produced by a format for reading data.

532 `Unreadable data encountered at record <nnnnnn>.`
This is usually produced by alphabetic data appearing where a number is expected.

533 `Warning: <nnnnnn> observations would not fit in data area.`
This is generated by APPEND when you are appending a large number of observations. This will usually result in the entire read operation being aborted.

534 `The last <nnn> variables specified would not fit.`
This is generated by APPEND.

535 `A name requested by READ is invalid.`
The name begins with a number, punctuation mark, etc.

536 `A name requested by READ is already in use.`
In this case, a new name, Xnnn is created. It is better to correct. Default names are difficult to keep track of.

537 `APPEND: No match found for <AAAAAAAA>.`
This is only a warning. The program is creating a new variable. You probably did not intend this.

538 `Warning: Name <AAAAAAAA> was not useable. Replaced with <AAAAAAAA>.`
See diagnostic 536.

539 `Variable list: The unidentifiable string is <aaaaaaaaa>.`
A list that is supposed to provide a set of variable names cannot be translated.

`Name range given, values are not low-high`
A command contains a list of variables AAAnnnn – AAAmmmm.

```
 Interaction Effects not allowed in this context
> 20 interactions in command - too complex. Table overflow.
> 50 interactions in table. Table overflow. Use DELETE
```

540
```
Variable list in command is > 150. Too large for models.
EXPAND(name) is not a valid form in this context.
```

541 `Only DSTAT and WRITE;list... may exceed 150 vars.`

542
```
Only one LHS variable for semiparametric estimator.
Panel data estimators not available for semiparametric.
Semiparametric estimator cannot use weights.
Semiparametric does not support heteroscedasticity.
Robust VC estimator is not available with semiparametric.
SCOBIT model not supported by semiparametric.
Semiparametric estimator cannot fit random effects.
Cannot use semiparametric for selection model.
Semiparametric cannot fit with choice based sampling.
Semiparametric cannot make out of sample predictions.
```
These diagnostics are produced by the semiparametric estimator for binary choice. They are produced by requests for unavailable options with this model.

`Number of GME support points must be one of 2,...,9`

543 `Kernel estimator needs LHS binary or proportions data.`

544 `Burr model is not available for panel data.`
`Burr model is only for binary choices.`
`Heteroscedasticity is not available for Burr model.`

545 `Random Effects Model requires panel data.  (LOGIT).`
`Random Effects Model requires individual data.  (LOGIT).`

546 `Fixed effects model is only available for binomial Y. (LOGIT).`

547 `Heteroscedasticity model is only for binary outcomes.`
`Heteroscedasticity model may not use panel data.`
`Selection model may not be based on het. logit.`
`Heteroscedasticity model is only for individual data.`

548 `Error: N not a multiple of T. N= <nnnnnnn>  T= <nnnn>.`
In a balanced panel, the full sample size must be an even multiple of the group size.

`Bad counter= <nnn>. (>200?) Row= <nnnn>,  group= <nnnnnn>.`
In an unbalanced panel, if a group size seems to be extraordinarily large, the program concludes that the counter variable is probably miscoded.

549 `Not enough workspace for <nnn> pds. and <nnn> vars.`
This should be unusual. It would happen with a huge model and large group sizes.

550 `Unable to create a new namelist for probs.`
The multinomial logit request for probabilities creates a set of variables.

551 `Not enough room in data for new variables (probs).`
The multinomial logit computation of probabilities creates several variables.

552 `Data area is <nnnnnn> by <nnnn> WKS cell is (<nnnnnnn>,<nnnnnn>).`
A cell in a worksheet file has a strange row, col index. *LIMDEP* cannot place the value.

553 `Same as 552 for placement into a matrix.`

554 `A label cell is not ascii text. Cannot read file.`
Something peculiar is in the first row of a spreadsheet file. Make sure it is just rows.

555 `Row index in XLS file > 65536. Cannot read it.`
These are bad data in a spreadsheet file. It should not be possible for *Excel* to write a row number larger than 65536 .

556 `Cannot fit exponential model for panel data.`

557 `Invalid setup (m1,m2) for generalized F model.`

558 `Variance het. model is for Logistic,Weibull,Normal models.`
The variance heterogeneity model is not available for the truncation model.

```
SURV cannot have both SELECT and HET/NORMAL
SURV with SELECT. Need to fit PROBIT;Hold model first.
This model is for MODEL=W, L, N, E, G or P only.
SURV with HET or SELECT does not support Variance Het.
SURV with HET or SELECT is not available for panels
SURV with HET or SELECT does not support truncation.
SURV with HET or SELECT does not support splitting.
Cannot fit this model with linear restrictions (;RST).
```

These diagnostics are produced by specification of a SURVIVAL model with heterogeneity.

559 `Data error. Individual <nnnn> T0=xxxx, T1=xxxx.`
This is a data error for one of the loglinear survival models with time varying covariates. T0 must be less than T1.

560 `Data error. Individual,<nnnnn>, has <nnnnn> records?`

561 `Data ended expecting,<nnnn> more lines for individual <nnnn>.`

562 `Gompertz model: Obs. <nnnn> duration < 0.`

563 `Data error: OBS.= I6,’ T= F8.2,’ Limits= 2F8.2).`

564 `Splitting model producing P=1 for all obs.`
This is probably the wrong model for these data.

565 `IF... or REJECT... Cannot ID string  <string is given>.`
This is probably a name in the string that is not in any of the tables (matrix, variables, scalars, etc.).

566 `1 x 1 data matrix in use is a missing value.`

567 `Missing values make matrices nonconformable.`

568 `Cannot resolve exponent in matrix power.`

569 `MATRIX Error is in  <string is given>.`

570 `Expected <nn> parameters for  <proc. name>. Found <nn>.`
Your EXECUTE command for a procedure has the wrong number of parameters in the list. The offending character string is then listed.

```
571   MINIMIZE/MAXIMIZE needs function definition with ;FCN=...

      Expected subfunction name to appear in FCN=...
      FUNCTION;... needs to include ;Keep=variable name.
      FUNCTION;...;Derivatives=namelist. Error in list name.
      Number of subfunctions must be less than 51.
      Invalid subfunction name <AAAAAAAA> is a (scalar, etc.)
      PARTIALS or DECOMPOSE;Function... must provide ;Labels
      FUNCTION;DERIVATIVES=namelist. Wrong # of names in list.
      Derivatives namelist contains ONE or name of KEEP var.
```

```
573   Variable name for quadrature equals a variable name.
      Variable name for quadrature equals a matrix name.
      Variable name for quadrature equals a scalar name.
      Variable name for quadrature equals a parameter name.
```
All these are different names from what was expected in this context.

```
574   GMM estimator unidentified: NPARM > # equations.
```

```
575   Conflict:param. and <AAAAAAAA> have the same name: <AAAAAAAA>.
```
You are using a label for a parameter in your optimization command that is already in use for something else, such as a scalar or matrix.

```
576   The variable in Plot(label) does not appear among ;Labels.
      CPLOT needs 2 labels in Plot(label1,label2) spec.
```

```
577   Observation  <nnnnnn>  variable  <AAAAAAAA> is missing.
```
This MATRIX operation does not bypass missing values. Use REJECT.

```
578   Incompatible B and RHS=list were given for kernel.
```
A simple count of elements in B and variables in Rhs does not match. It looks like B is from a different model.

```
579   Sample size= <nnnnnn> is too <large or small> for kernel estimator.
```

```
580   Error occurred attempting to open file for MGET/MPUT.
```

```
581   Error occurred in READ/WRITE during MGET or MPUT.
```

```
582   Found [ELSE], but no prior IF[...] was set. (MATRIX command.)
```

```
583   Problem with matrix. See previous diagnostic.
```
This is generic. There are several possible conditions. A previous diagnostic will indicate the problem.

```
584   Expected number in Quad(points,type) not found.
      Expected L or H in Quad(points,type) not found.
      Quad points not one of the available set.
```
This is just a request to list the weights and nodes for quadrature.

585 `Total cells= <nnnnnn>, Space= <nnnnnn>, Cells needed = <nnnnnn>.`
The result of a MATRIX command is too large. Maximum size is 22,500 cells.

586 `Estimated E[y1|y2,...]=0. Cannot compute partials.`
Marginal effects for the multivariate probit model produce an unusable numeric result. *LIMDEP* is unable to continue.

587 `You did not define EQ<n> for  <AAAAAAAA>.`
This is the multivariate probit. You have fewer equations defined than there are Lhs variables.

588 `Error translating function in optimization command.`
The offending character string will be listed. This is usually caused by an unrecognized name in the expression. This will be listed in a previous error message.

589 `Error is in  <character string is listed>.`
This is the translation error in WALD or an optimization command. A previous diagnostic will show the cause of the problem.

590 `Obs.= <nnnnnn>  Cannot compute function:  <a hint is given>.`
This is produced by the optimization programs. It is data dependent. The hint may be able to show the source of the problem, for example zero divide, number too large for gamma function, etc. You must examine the data to determine the exact cause. The observation that produced the error is given.

591 `Cannot censor data with R.E. Model.`
The ordered probit model with random effects cannot also accommodate censoring. This is for cross section data only.

```
HIOP model does not allow censoring.
HIOP model does not allow panel data treatments.
HIOP model does not allow selection or zero inflation.
HIOP model requires HI1=list or HI2=list, not incl. ONE.
```
These diagnostics relate to the hierarchical ordered probit model.

592 `Censoring indicator <AAAAAAAA> = <xxxx> at obs <nnnnnn>.`
The censoring indicator for the ordered probit model is supposed to be binary.

593 `Not enough uncensored observations to continue. (OProbit.)`

594 `Stratum <nn> has no observations in it. (Ordered probit.)`
LIMDEP cannot estimate the threshold parameters for this stratum.

595 `Warning, NPRD= <nnn>,  P(i) may be inaccurate.`
When you compute a random effects model, the group probability equal to the product of the member probabilities is computed, not the sum of logs. If you have many periods, this becomes too inaccurate to compute the log likelihood with it.

```
Hessian is singular at rho=0. Cannot compute LM test.
```
Problem with random effects probit or ordered probit.

596 `Error: N not a multiple of T. N= <nnnnnn>. T= <nnnnnn>.`
For a balanced panel, the full sample size should be a simple multiple of the group size.

`Bad counter=<nnnnnn>.(>75?)Row= <nnnnnn>, group= <nnnnnn>.`
The counter variable takes a peculiar value, nonpositive or unexpectedly large.

597 `Not a matrix save file!  MGET request uses the wrong type of file.`

598 `You must provide ;NOBS=value to read into matrix.`

599 `Nobs*Nvar > maximum matrix size of 22500 cells.`

600 `Simplify expression or separate WALD functions.`

601 `Obs: <nnnnnn>  <AAAAAA> = <xxxxxx>. Gradient not computable.`
The MLE for the Box-Cox model requires all data to be transformed to be nonnegative. The transformation can be computed even for zero, but the derivatives of the log likelihood as well as the correct asymptotic covariance matrix require all data to be transformed to be positive.

602 `Error attempting to convert date: <date string given>.`
This is an internal error that should not occur. You may be using a date in a CREATE function that is out of range. For example using yearly data from 1960 to 2000, and creating with the function Ind (2001) might cause this error since this date is out of range.

603 `PDS= <nnnnnn> NOBS= <nnnnnn> Panel is not balanced.`
Your REGRESS command suggests the panel is balanced with ; Pds = value, but the sample size is not a multiple of the indicated number of periods.

604 `No room in data area to create group indicator. (REGRESS ; Panel)`

```
The nested random effects model is only for regression.
Use only ;STRATUM and ;CLUSTER for the nested RE model.
Nested random effects model must be unconstrained.
Nested random effects model does not support AR(1).
Nested random effects model does not support weights.
Cannot fit nested RE model with random parameters.
Nested RE model cannot do Murphy Topel 2 step VC.
```
These diagnostics all relate to the panel model with nested random effects.

605 `PDS variable has missing values. Cannot continue.`
`STR variable has missing values. Cannot continue.`

606 `Stratum <nnnnnn> has > 20000 obs. Is this a panel?`
This does not appear to be a panel.

607 `Too many groups for noncontiguous panel data.`
When data are arranged haphazardly in the sample, if the sample is small enough, it is still possible to create the counters and pointers needed for the regression. If N is too large, however, estimation is not possible. Try sorting the data so they are more convenient.

608 `AR(1) model cannot be computed with weighting.`

609
```
Bad setup for Haus/Tay estimator.kx1+kx2+kz1+kz2 not =#rhs
Hausman/Taylor estimator: KX1 must be positive
Model is not identified. KX1 must be >= KZ2.
Hausman/Taylor. Must provide s2e,s2u, both positive.
```

610 `A valid observation has a missing PERIOD variable.`

611 `Outer strat. var takes value > 10000. Recode.`
This is the two way stratification in the random effects model. The outer stratification variable must be a complete sequence of integers.

612 `Number of bootstraps must exceed K for VC matrix.`

613 `(N,K) must not exceed (5000,15) for LAD estimator.`

614 `# RHS variables may not exceed 100 for ICLS.`

615 `NOBS may not exceed 100,000 for ICLS.`

616 `Error in specification of equality constraints.`

617 `Lower bound for X( <nn>) is greater than upper.`
Your linear programming problem is not set up properly.

618 `Solution vector has infinite components.`
There is no solution for the linear programming problem.

619 `No feasible solution exists.`

620 `Maximum iterations exceeded.`

621 `Numerical instability.  Cannot solve problem.`

621
```
Panels for Malmquist indices must be balanced. Fixed T.
Cannot bootstrap Malmquist indices. NBT=# is ignored.
Cannot compute allocative inefficiency for Malmquist.
Number of prices supplied must equal number of inputs.
ONE is not a valid OUTPUT (LHS) variable.
ONE is not a valid INPUT (RHS) or PRICE (RH2) variable.
DEA is limited to 16,000 observations (firms).
Peers tabulation is only for small samples - N <= 1000.
Panel data must be balanced, no missing, no zeros.
Number of bootstraps must be between 10 and 1000.
Not enough workspace for the number of bootstrap reps.
Solution vector has infinite components. No soln.
```
These diagnostics are checks on the data envelopment procedure in FRONTIER.

622 `Inconsistency found in constraints.`
This is a problem in setting up a linear programming problem.

623 `Check for error in  <string is given>.`
`Look for: Unknown names, pairs of operators, e.g., */`
This is a general diagnostic for the CREATE command. Something in an expression could not be identified. A character string that contains the error will be included in the diagnostic also. Note, during compilation, strings in parentheses are reduced from the inside out, and internally, the reduced string will be given a symbol with a lower case letter. Thus, the string from X=(A+1)*/(C+D) (which contains an error) might appear in the diagnostic as 'aa*/ab' where the 'aa' is the symbol for (A+1), etc. The internal coding with lower case letters is used to break down expressions in parentheses. The first letter indicates the subcommand – subcommands are separated by semicolons. 'a' is first, 'b' is next, etc. The next letter indicates the expression in parentheses in the order encountered. Thus, 'ab' is the D+1. Unfortunately, when there are nested parentheses, this can become confusing, since the listing just continues in order. Thus, the final expression in ((A+1)*/(C+D)) is 'ac.'

624 `No valid data found in sample=1 to <nnnnnn>.`
A SAMPLE command, SAMPLE ; n1 - n2 $ produced a sample with no valid data.

625 `2 way F.E.M. No observations in period <nnnn>.`
The period dummy variable coefficient cannot be estimated. (Column of zeros.)

626 `Insufficient degrees of freedom for group means regression.`

627 `You must fit the PROBIT or LOGIT model to use ZIP.`
This is for the Nagin and Land model for counts.

628 `A variable name appears more than once in list.`
This is the setup for the Nagin and Land estimator for the Poisson model.

629 `GAMMA count model is unstable when Vy/Ey > 10.`
LIMDEP terminates. You might try the negative binomial model.

630 `Error: N not a multiple of T. N= <nnnnnn>. T= <nnn>.`
This is for the count data model for panel data. You are using a balanced panel, but the sample size is not a multiple of the group size.

631 `Bad counter= <nn>. Row = <nnnnn>  group= <nnnnnn>.`
The group size for a panel data count model took a nonpositive value. This occurs during a data check.

632 `Counter variable contains an error. Check values.`
In the count data model for an unbalanced panel, it looks like the counter is out of sync. The count took the observations for a group past the end of the current sample.

633 `Negative LHS value found. Cannot fit the Poisson (count) model.`
`Variable  <AAAAAAA> is always zero.`
`No variation observed in <AAAAAA>.`
`Singular Hessian in  <AAAAAA> model for  <AAAAAA>.`
`Maximum iterations fitting  <AAAAAA> model for  <AAAAAA>.`
The program that produces this diagnostic is used to compute starting values for Poisson, probit, logit, and several other models. The diagnostic is specific to the kind of model being computed.

Diagnostics 634 - 637 are produced by the Arellano and Bond, dynamic panel data estimator, called by REGRESS ; Panel ; Dpd ; ...

634 `Unable to read your ;START=LIST specification.`
`4 values given in ;START must sum to number of RHS variables.`
`4 values given in ;START must be nonnegative.`
`;START=list must give KX1,KX2,KF1,KF2[,s2u,s2e].`
`Values given for s2u and s2e must be positive.`

635 `Your model is too large for this pgm. (> 50 RHS vars.)`

636 `This estimator is limited to T(i) <= 100 periods.`
`You have too many instruments (moment conditions).`
`Invalid values given for First date - Last Date in DATE=...`
`DATE=variable for DPD must give a sequence of integers`
`Date given is not within First - Last as in command.`
`DPD: Looking for DATE=Variable,first date - last date.`
`DATE variable given for DPD/Panel is not in names table.`
`Unable to construct model for ZTYPE=P and unbalanced panel`

637 `Unable to invert moment matrix for model VC matrix.`

Diagnostics 638 – 641 are produced by the random coefficients model.

638 `Last group: <nnnn>  ID= <nnnnnn>. Only 1 observation.`
`The random coefficients model cannot be computed.`
`i= <nnnn>, index= <nnnnnn>,  T(i)= F6.1,' too small.`

639 `i= <nnnn>, index= <nnnnnn>, singular moment matrix.`

640 `i= <nnnn>, index= <nnnnnn>, Perfect fit, s^2=0.`

641 `Not enough groups (only <nnn> ) to fit model.`

642 `Heteroscedasticity model does not support control variable.`
This is only for the sample selection model.

643 `Heteroscedasticity model does not support instrumental variables.`
This is only for the sample selection model.

644 `Cannot select on  <AAAAAAAA> =  <nnn>.`
Sample selection model with multinomial logit selection equation. You used ; Choice = jj, but the Lhs variable in the logit model does not take this value.

645 `Hausman and Wise attrition model must be preceded by PROBIT;HOLD...`
`Starting value for SIGMA is not positive.`
`Start values for r12 and r23 must both be in (-1,1).`

646 `Starting value for RHO must be in (-1,+1).`
`Staring value for SIGMA must be positive.`

647 `Label <AAAAAAAA> is used twice in procedure.`
`The parameters in a procedure must each have a unique label.`
`Loop index <AAAAAAAA> used in more than one DO.`

648 `Execute changes a protected scalar, S etc.`
EXECUTE ; Name = values $ cannot change the scalars S, RSQRD, etc.

649 `Loop index <AAAAAAAA> was used in EXEC command.`
Your DO loops must use different index names from the EXECUTE command.

650 `ENDDO; <AAAAAAAA> $. No matching DO found.`

651 `DO...; <AAAAAAAA>;...$ No matching ENDDO found.`

652 `GROUPED: Obs= <nnnnnn> Limit values are not ordered.`
If the limit values are constants, you can see the problem in the command. But, if the limit values are given by variables, you must look at the data to find this problem. The observation number is given for this reason.

653 `Marginal effects, stratification variable not in table.`
; Margin = name for stratification gives an unknown name.

654 `Too many strata in marginal effects setup. > 9.`

655 `Marginal effects, Variable = <AAAAAAAA> stratum <nn> is empty.`

656 `EXEC;WHILE or UNTIL... Condition fails on the first try.`
`Cannot get started.`

657 `Invalid value (K > 0 to 15000) given for ;DRAWS=Nb.`

658 `Error encountered translating ;LCM=list.`
The problem is an unknown variable name.

659 `Maximum of 10 variables allowed in ;LCM=list.`

660 `Not Used.`

Diagnostics 661 - 663 are produced by the WRITE command.

661 `Problem with format given for WRITE command.`

662 `Cannot process internal ( ) or / in [format].`
`Did not find expected ending ] in format.]`
`Format type is not X, F, I, D, or E.`
`I, F, D, or E format types need field with specification.`
`No .d specification was given for D, E, or F format.`
`The .d format is not used for X or I format.`
`X format code does not allow w.d in spec.`
`Error in repeat count of format specification.`
`Error in w part of w.d spec. in format code.`
`Error in .d part of w.d format specification.`
`Dw.d/Ew.d requires w-d > 5, Fw.d, w-d > 1.`
`Format does not give enough codes for variable list.`

663 `Cannot APPEND. You have not opened a file.`
There can only be one line per observation. The WRITE command writes to the screen when you do not give a file name. This is an error in the WRITE operation. Check the FORMAT or file specification.

664 `No observations in stratum <nn>.`
This is the loglinear survival model or the Cox model. A stratum is empty.

665 `Frontier with ;HET must give ;HFv and/or ;HFu.`
`You must fit PROBIT ; Hold before FRONTIER;SELECT'`
`Battese-Coelli model requires a panel data set.'`
`Battese-Coelli model does not support hetero.'`
`Scaling model requires ;RH2=list. (Else use ;HET;HFu &/or v)`
`Scaling model requires ;RH2=list of variables (not ONE)`
`Scaling can only have HFU. s(v) is a constant.`
`SC form of frontier scaling requires ;HFU=list`
`Basic SF model was inestimable. Trying panel model.`

666 `Str variable= <AAAAAA>  Obs.= <nnnnnn>,  value= <nnnn> > 100.`
This is the stochastic frontier model using panel data. A stratum has too many observations.

`Stratum  <nnn> is empty. Unable to continue.`
This is for the frontier or limited dependent variable with panel data.

`Use REGR;Lhs=one ;Rhs=one;Str= <AAAAAAAA>$ to create _STRATUM.`
This is a way to compute a correctly created stratification variable for the panel data estimators.

667 `Warning, NPRD= <nnnn>; P(i) may be inaccurate.`
To compute the log likelihood it is necessary to compute the product of the densities for all observations in the group. If the group size is large, this is likely to be too small to compute accurately.

668 `Error: N not a multiple of T. N= <nnnnnn>. T= <nnnn>.`
In a balanced panel, the sample size must be a multiple of the group size.

669 `Bad counter= <nn>. Row= <nnnnnn>,  group= <nnnnnn>.`
The count variable for a panel takes an unusable value, such as zero or negative.

670 `Variable  <AAAAAA> has no variation. No regression.`
The error occurs computing a least squares regression, possibly for many different models.

671 `GARCH model with Q=0 is fit by OLS.`

672 `Nobs is < # groups. Cannot compute full GLS.`
In the TSCS model, the number of periods must be larger than the number of groups. If not, then Sum(ei ei') does not have full rank, and GLS cannot be computed.

673 `TSCS: Error reading ;GROUP=list...`

674 `Cannot compute TVC <nn>, Observation=<nnnnnn>.`
This occurs during estimation of the Cox model. It is data dependent, so the observation is given.

675 `Found nonpositive LHS variable in loglinear model.`
`Data for SURV:inv.gauss must be in log form. Are they?`
`Found value of Y outside (0,1).`
These are checks on the values of the dependent variable for the beta regression, gamma regression, exponential regression, or inverse Gaussian regression.

676 `Fixed effects model requires ;PDS=specification.`

677 `Counter <=0: <nnnnnn>. Row =  <nnnnn>,  group= <nnnnn>.`
This occurs during a fixed effects model. There are many different models in the ; Fem group. This occurs during a data check on setup for the panel.

678 `Error: N not a multiple of T. N= <nnnnnn>. T= <nnnn>.`
In a balanced panel, the sample size must be a multiple of the group size.

679 `Obs. <nnnnnn> is < 0. Cannot fit Pois/NegBin.`
`Obs. <nnnnnn> is <= 0. Cannot fit loglinear model.`
`Obs. <nnnnnn> = xxxxxx FEM/probit mdl needs 0/1 or proportion.`
All of these are data checks on appropriate values for the Lhs variable while fitting a fixed effects model.

680 `Latent Class Model requires a panel to be estimated.`
You specified ; Lcm without specifying a panel with ; Pds = spec.

681 `Did not find positive values of lambda and sigma.`
LIMDEP is looking for starting values for the panel data version of the stochastic frontier model. You must precede the panel data version with an identical cross section model command.

```
682   B vector is too short. Did you fit frontier for B0?
      Expected to find B from previous FRON..;PAR command.
      Starting value for LAMBDA=su/sv is not positive.
      Starting value for SIGMA=sqr(s2u+s2v) is not positive.
```

LIMDEP is looking for starting values for the panel data version of the stochastic frontier model. You must precede the panel data version with an identical cross section model command.

```
682   This model must be fit first w/o latent classes.
      This model must be fit first w/o RPM specified.
      Model spec. must be identical to the initial estimator.
      Starting value for LAMBDA=su/sv is not positive.
      Starting value for SIGMA=sqr(s2u+s2v) is not positive.
```

These diagnostics are produced by the Battese and Coelli frontier estimator.

```
683   B vector is too short. Did you fit zip model for B0?
      ZAPTAU not found or wrong size. Did you fit ZIP mdl?
      Expected to find estd. ALPHA. Did you fit ZINB mdl?
```

LIMDEP is looking for starting values for the panel data version of the ZIP model. You must precede the panel data version with an identical cross section model command.

```
684   GROUPED/panel must be preceded by GROUP;Rhs=...,one...$.
      GROUPED requires internal limit values with ;LIMITS=
```

```
685   This estimator does not allow ;RST or ;CML.
```

```
686   Starting value for theta in negbin must be > 0.
      Starting value for sigma must be > 0.
      Starting value for rho must be in (-1,1).
```

All of these are checks on an ancillary parameter for a panel data model. This will generally not occur if the internally generated starting values are used, but can occur if you supply your own.

```
687   Your TIME variable exceeds maximum period from PDS.
```

```
688   Iterations aborted by user request. Exit status=-1.
```

```
689   Exit from iterative procedure. <nnnnn> iterations completed.
```

```
690   This model is inestimable if parms. are correlated.
```

The random parameters stochastic frontier model allows parameters in the mean, the mean of the truncated distribution and the variance all to be random. They cannot be correlated, however.

```
691   Biv Prb needs y1,y2, RH1 and RH2. MDL is incomplete.
```

Your bivariate probit random parameters model command is incomplete.

```
692   Bivariate Probit needs y1,y2, RH1 and RH2.
```

Your model command is incomplete.

693 `For ;MEANS, need ONE in both SELECT and PROBIT.`
`In SelectREM, both equations must contain ONE.`
This is the panel, random parameters (Zabel) form of the sample selection model.

694 `FCN for RP model is variable(t) or [t]. t=N,T,or U.`
`Did not recognize variable name in FCN setup.`
In the ; Fcn = (name) or [name] specification, the name must be included in the Rhs variables.

695 `Variable in FCN=name ( < or [ type ] > or > is not in RHS/RH2/HFN list.`
Note that if a variable is in an Hfn list, it will usually appear as [name] not (name).

696 `There are no complete obs. in the data set!`
If you have a panel that has missing values in every group, the model will be inestimable.

697 `Previous SELECT model does not match. Was it MLE?`
`Did not find sigma>0 and -1<rho<1 in previous MLE.`
SELECT ; Rpm ; ... must be preceded by an identical MLE for the model to produce starting values.

698 `Unsuccessful initial estimation to get start values.`
This is produced setting up the random parameters, fixed effects, or latent class models for any of the model types, probit, tobit, etc. If this diagnostic occurs for a cross section version, then the panel version will not be estimable either.

699 `GROUPED requires internal limit values with ;LIMITS=.`

700 `GROUPED/panel must be preceded by GROUP;Rhs=...,one...$.`
The panel data version requires an identical cross section version to provide the starting values.

701 `Starting value for RHO must be in (-1,1).`
This is produced while reading your starting values for a panel data version of the bivariate probit model.

`Start value for RHO must be in (-1,1).`
This is produced while reading your starting values for a panel data version of the sample selection model.

702 `Starting value for sigma or theta must be > 0.`
`Start value for sigma, theta (or lambda) must be > 0.`
This is produced while reading your starting values for a panel data version of the tobit or negative binomial or other limited dependent variable model.

703 `Cannot list more than 10 outcomes.`
The panel data models with ; List are limited to group sizes of 10 or less.

704 `Panel 2sls, # INST vars. must be >= # RHS vars.`
`Panel 2sls may have only one of ;FIX,;RAN,;DIF,;MEA`
`Panel 2sls, Too few observations to compute estimator`

705 `NOTE: Analytic Hessian is not PD. Using BHHH for variances.`
The latent class models compute the analytic second derivatives matrix for all models. If convergence was not very close to the true function maximum, this matrix may not be positive definite. If so, the BHHH estimator is used instead.

706 `Demonstration version does not support SAVE.`
`NLOGIT_ACA can only read its own project file.`

707 `Demonstration program can only load its own file.`

708 `SAVE file is badly constructed. Cannot read it.`

709 `Data array in file is < Nmax*Kmax. Cannot store.`

710 `Unable to load project. Restarting. NKMAX=200000.`

711 `CREATE  Variable in EXPAND is not an integer from 1 to 100.`

712 `EXPAND for  <AAAAAAAA> needs <nnn> variables. Only <nnn> are available.`

713 `You already have 10 namelists defined. Cannot EXPAND.`

714 `Variable specified for recode is not in NAMES=...`

715 `Both LHS variables are dichotomous. Use BIVARIATE PROBIT.`
`Noninteger value found for one of the LHS variables.`
`Unable to display frequency table. Both dimensions > 8.`
`One of the LHS variables takes a value > 20.`
`No starting values were provided.`
`Unusable starting vector. Needs B1,MU1,B2,MU2 (RHO opt.)`
`The wrong number of starting values was found.`
`Absolute value of start value for RHO is greater than 1.`
`Exiting from function computation with error.`
`Check setup for ;KEEP = name1,name2. Found an error.`
One of the outcomes almost never occurs. Estimation of the threshold parameters will not be possible. These diagnostics relate to the bivariate ordered probit model.

716 `In MATRIX ^ power, MATRIX is not square.`

717 `MATRIX^matrix power. Not same dimensions.`

718 `Matrix^power. Exponent is too large, > 10.`

719 `Cannot raise non positive definite matrix to negative power.`

720 `Raising negative root to noninteger power.`

721 `Raising large root to power causes overflow.`
Overflow occurs when a number becomes too large for the computer. This is approximately exp(638).

722 `Raising negative value to noninteger power.`
A matrix is being raised to a power.

723 `Raising large value to power causes overflow.`
Matrix being raised to a power produces a large number being raised to a large power.

724 `Expected no more than 5 specifications in ;SDV=list of 1 or *s.`
This is for simulation estimation by MAXIMIZE. You may either allow free or unit constrained variances for the simulation variables.

725 `You must specify ONE of ;PDS or ;STR for a panel.`

726 `Reordering can only be done for balanced panels.`
Your REGRESS ; Panel command requests that the observations be reordered. This can only be done for a balanced panel.

727 `Can only reorder up to 100,000 observations.`

728 `<nnnnnn> is not enough complete observations to continue.`
You have too many missing values in the data set to fit the model you have specified.

729 `Unable to optimize function. Collinearity?`

730 `Cannot optimize. Constraints are inconsistent.`

731 `Variable <AAAAAAAA> always = <xxxxxx>. No variation!`
This applies to independent variables in a model. This is checked at the time an OLS regression is computed, usually for starting values. If it occurs, your model will not be estimable.

732 `Warning. OLS gives a perfect fit.`
This shows up in many possible contexts. It usually means that the model you want cannot be computed.

733 `Cannot fit exponential survival model for panel data.`

734 `SELECT AND BPROBT  Check setup for fixed effect SELECT model.`

735 `Tobit and Poisson/FEM cannot keep cprobs.`

736 Latent class form is not available for this model type.

The ;RST=list specification is not available for RPM.
The ;CML=list specification is not available for RPM.
R.E. is not supported for latent class models.

Latent class form is not available for this model type.
The REM=spec... is not supported for this model.

Cannot HOLD probit results from an R.E. model.

737 Selection RP model requires PROBIT;...;HOLD as usual for selection models.

738 Error in ;TABLE = MODELNAME. Check syntax.

739 ;KEEP/RES/PROB/GROUP/CPROB, expected = name not found.

740 ;KEEP/RES/PROB/GROUP,CPROB=name, name is too long.

741 ;KEEP/RES/PROB/GROUP,CPROB=name, name is invalid.

742 Cannot identify specification given for ;PDS=....

743 Cannot identify specification given for cluster.
This needs ; Cluster = number or ; Cluster = a variable name to give the stratification.

744 Invalid value given for quadrature nodes. Ignored.

745 Panel estimators (;PDS=...) do not support ;CLUSTER=....

746 Cannot use both WTS and CLUSTER in fitting a model.

747 Sample is too long to sort. N= <nnnnnn>.
The limit is 100,000 observations.

748 ;SIGMA=NAME, sigma is supposed to be a matrix in this context.

749 ;sigma=NAME only used for SURE,3SLS,NLSURE,GMME.

750 ;REG/SPIKES/BARS/GRID/NOFILL/BARS only for PLOT and MPLOT.

751 TIME=variable. Variable given not recognized.

752 LIMIT=name or value. Name not recognized or bad value.

753 ;LAGS=value for LSDV estimator. Could not read value.

754 ;NBT=value (bootstraps). Could not read value given.

755 `;RHO=value is only for REGRESS and 2SLS.`

756 `DSP=value (Negbin) or ;THETA=value (Box-Cox).`
The value given is not positive.

757 `; Hold(IMR=...) No room to create IMR variable. Data area is full.`

758 `;Spikes or ;Bars=list. Could not read list of values.`

759 `Name in expr not label,variable,scalar,matrix,list,or FNi.`
This is generated by optimization or WALD.

760 `Optimization or WALD. Expression has unmatched square brackets.`

761 `Found \, expected \number\ did not find it. (Optimization or WALD).`

762 `Obs.<nnnnnn> of <AAAAAAAA> = <xxxxx>, not 0 or 1.`
Data for the multivariate probit model must be binary.

763 `Cannot identify namelist in ;UTILITY=namelist.`

764 `UTILITY=name must specify a namelist of NY names.`

765 `The namelist in UTILITY=name may not contain ONE.`

766 `Did not find a valid name in UTILITY=name.`

767 `CREATE, name in expr is not var.,scalar,matrix, etc.`

768 `Have been unable to break down the command segment.`
Diagnostic 623 will follow this diagnostic and show the function in question.

769
```
You must provide ;TRIALS=spec. for binomial model.
Fixed number of trials must be a positive integer.
Invalid missing value found for LHS variable.
Invalid missing value found for number of trials.
LHS variable for binomial model must be integer >= 0.
Number of trials for all obs. must be integer > 0.
Bad data found. Y must always be <= number of trials.
Did not find any nonzero values for LHS variable.
LHS variable always takes the same value.
```
These are all checks on the loglinear, binomial regression model.

769
```
Invalid data for geometric model. Must be 0,1,2,...
Invalid data. Mean of Y must be positive.
```

```
770   MATCHing command must be preceded by PROBIT or LOGIT ;Hold.
      MATCH can only analyze up to 200,000 observations.
      ONE is not a valid outcome variable.
      Invalid treatment dummy: Obs/Row=<NNN>/<NNN>= <NNNN>.
      <NNN> valid obs. is too small for matching analysis.
      There is no variation in the outcome variable. Check data.
      There are no control observations in the sample.
      There are no treated observations in the sample.

771   LOGIT/RPM can only fit with up to 20 outcomes.
      Data for LOGIT/RPM are not coded 0,1,...,J up to 19.
      LHS variable for LOGIT must contain all values 0...J.

772   ;Labels=list. Expected <NNN> names. <NNN> were given.
      CLASSIFY: limit is 1,000 distinct groups.
      CLASSIFY: Exceeding workspace. Too many variables.
      CLASSIFY: Bad values given for priors.
      CLASSIFY: Wrong number of priors given.
      CLASSIFY: wrong number of labels given.
      CLASSIFY: Singular covariance matrix for a group.
      CLASSIFY: Priors must be between 0 and 1.
      Priors for classes do not sum to 1. Check values.

775   ;CLS can only be done once per command. | is not available
      Cannot read value after = sign in test specification
      Variable name in test specification is not recognized
      Variable in test spec. does not appear in the model

776   Missing data reduce sample to 2 or less. No plot!

778   WARNING. N*(K+1) > 50000. Cannot store matrix beta(i).
      Unable to obtain MLE of lambda for frontier model.
      Estimated lambda < 0. Cannot estimate inefficiency.
      LOWESS regression is limited to 5,000 observations.
      LOWESS can only save predictions for one LHS variable
      LOWESS must be based on a nonconstant x, not ONE.
      Local linear regression is limited to 20 regressors

779   Unusable starting values. Cannot continue.
```

This is produced by the random thresholds ordered choice model.

```
780   Invalid name specified in LOCAL declaration
      Name in LOCAL declaration is a reserved <AAAAAAAA> name.
      LOCAL;type=list$ Type must be matrix, scalar or variable.
      Insufficient work space to set up local variable.
      LOCAL;=list...$ Did not find equals sign.
      LOCAL;type=list.... Type appears more than once.
      LOCAL;type=list. Too many (> 10) items in list
      Name given in LOCAL;type=list is > 8 characters.

781   The stratification variable must be an integer.
      Can only tabulate up to 99 strata
      Found only one stratum in sample. Use DSTAT
      Found nonpositive value for weighting variable.
```

```
782   Over 100,000 values for Sqq(.) function. Overflow

783   BINARY CHOICE ANALYSIS requires ;Start = values.
      Number of start values must match # vars. in RHS.
      Model must be LOGIT, PROBIT, GOMPERTZ or COMPLOG.
      Did not find */-+ or > in scenario specification.
      Unknown variable name in scenario specification.
      Variable in scenario does not appear in the RHS.
      Bad value found in scenario specification after = sign
      FORMAT('Cannot compute ROC for > 375,000 points.

783   Name in ;IMPUTATION=name is not in list w/ EXEC
      ;Imputation is not useable with LAST MODEL
      ;Imputation is only useable inside a procedure.
      LastModel command missing ... = specification.
      Unable to read parameters or covariance in LastModel
      Number of labels does not match number of parameters
      Covariance matrix is the wrong size for parameters
      Covariance matrix is not symmetric
      Covariance matrix is not nonnegative definite
      LastModel must include ;Parameters and ;Covariance

784   DECOMPOSE must be preceded by model setup
      Sample size too small for decomposition
```

Diagnostics 785 are produced by the PARTIAL EFFECTS and SIMULATE commands.

```
785   SAVE cannot be used with ;Means.
      Cannot locate X variable for effects
      | variable W = ... is badly specified.
      | variable W ... cannot find the var.
      | variable W = list...  Bad list.
      & variable Z = ... Cannot find Z.
      & variable Z = L(D)U. Bad list found.
      Insufficient observations to get APE.
      Nonpositive increment D in L(D)U
      ;PLOT requires & Z = L(D)U in spec.
      Not enough room in data area to SAVE.
      Cannot compute elasticities for binary X
      ;Effects: ONE ... is not a valid spec.
      ;Effects:...@ D. Cannot find D in table
      in @D=values, unreadable or > 10 values
      In @D=values or @D, not a set of integers
      ;Fix:variable =...Cannot identify variable
      ;Fix:variable=value. Cannot read value
      ;Fix:variable... Variable cannot be fixed
      Cannot do sample splitting if not ;Average
      Conflict between ;Fix=... and scenario
      ;Outcome=value in PARTIALS. Value is unreadable or < 0
      Error in construction FN*=label*vector(j1:j2)
      No model has been stored yet.
      PARTIALS or DECOMPOSE w/ ;Function... must provide Labels'
```

```
786   Did not find list to LIST or DELETE in table.
      No equals sign found in LABELS command.
      LSTMODEL is reserved for program use. Change the name.
      25 label lists already exist. No room for a new one.
      Label list <AAAAAAAA> is full. Unable to store

787   You must open an EXPORT file before exporting matrices
      Matrix <AAAAAAAA> has > 255 columns. Cannot export it to Excel.
      You must OPEN;EXPORT=file before exporting results

788   Block diagonal matrix exceeds 223x223
      Matrix in BLKD list is not in matrix table.
      Matrices in block diagonal list must be square

789   Y must take values 0,1,2,3,4 for GHH SAH model
      RHS for this model must be ONE,...
      Did not find namelists XR and XM for GHH model
      First name in XR and XM namelists must be ONE.
      Did not find right start values from OPROBIT
      Did not find start vector MU with 3 values
      Did not find start vector BR with 1+kr values
      Did not find start vector BM with 1+km values'

790   Singular moment matrix for Xj, X or Yj. Check model spec
      Problem with model spec. Smallest root is < or = 1.
      Y cannot be ONE in this LIML context.
```

These diagnostics are produced by the LIML estimator for linear models.

```
791   All RHS variables are in INST list. Cannot test.
      Not enough instruments provided for Wu test.
      Insufficient observations in sample for Wu test
      Instruments are collinear. Cannot compute Wu test.
      Instruments and X are collinear. Cannot do Wu test
```

These diagnostics are associated with the Wu test for 2SLS.

```
792   No within groups degrees of freedom. N groups!
      There is no within group variation in this X.
      No between groups degrees of freedom. 1 group!
      No valid observations in the sample!
      No variation in this X! All values are identical

793   There is no variation in the LHS variable.!
      In truncation/endogenous model, y must be > 0.
      Exposure var: <AAAAAAAA> = <xxxxxxxx> at row,obs=<nnnnnnnn>

794   RMN(mu,V), unrecognized name given for mu and/or V
      RNM(mu,V), more than 2 matrices given in the list
      Nonsquare V matrix given for RMN function.
      You must give a vector for MU in RMN(mu,V)
      Nonconformable vector MU and matrix V in RMN(MU,V)
      Maximum size of created sample in RMN is 100 variables.
      You already have 25 namelists. Unable to create a new one.
      No room left in data for new variables from RMN
      Matrix V given in RMN(mu,V) is not symmetric
      Matrix V given in RMN(mu,V) has a negative root.
```

```
798   Interaction variable no longer exists. Last model is unuseable.
      Eqn. kept with ;HOLD no longer useable. Int.varbl missing.
      Namelist <AAAAAAAA> is no longer useable. Int.
      IMPUTE;Lhs= . . . must precede EXEC;Imputation ...
      TYPE must be M, B, F, C, T or O in IMPUTE command
      IMPUTE must contain LHS, RHS and Type specifications

801   LHS var in IMPUTE;fill may not be in an imputation model

801   <Model command> is not used with RPMAX/RPMIN
      RPMAX/RPMIN must provide <Specification>
      RPMAX/RPMIN is not a <Model type> command')
      No derivatives in RPMIN/RPMAX command or ;FCN=...
      Use label[value] to fix parameters
      No multiple equations spec. in RP...
      Start values are given in ;Labels=...
      Panel data are not set up correctly.
```

The RPMIN and RPMAX estimators have very specific requirements for the model specification.

```
802   Create ; name = stk(...). Did not find equals sign.
      Expected to find STK(..). Did not find ( or ).
      Processing STK function. Did not find expected , / or )
      Cannot make NAME=Stk(... same NAME ...). Change LHS.
      Cannot identify item ',8a1,' in STK(...list...)
      Rows in stacked matrix define different #s of cols.
      Stacked matrix has too many rows for your data area.
      Stacked data matrix is too large; > 250,000 cells.
      You do not have room to create a new variable.
      You have already defined 25 namelists. No room left.
      You do not have room to create a new variable.

802   GLIM  command must contain ;MODEL=type.
      Unrecognized model type in GLIM;Model=type.
      You must provide ;TRIALS=spec. for binomial model.
      Fixed number of trials must be a positive integer
      FEM/probit mdl needs 0/1 or proportion

803   Error encountered translating ;LCM/DCEVM=list.
      Maximum of 30 variables allowed in ;LCM/DCEVM=list.
      Unable to identify name in ;CLASSP=namelist
      Cannot save LC parameters in ONE. Use a new variable.
      Unable to identify name in ;PAR=namelist
      CLASSP=list contains ONE. Cannot save P(j|i) in ONE.
      ;PAR=namelist(lclist). Error in lclist.
      ;PAR=namelist(lclist).Lists must be same length.
      ;PAR=nlist(lclist). Vars. in lclist must be in nlist

804   Setup for conditional model command has an error.
      Model ; (scalar reln value) ... $. Scalar unknown
      Model ; (scalar reln value) ... $ Bad value given.
```

```
805   NEW);namelist=list redefines an existing namelist
      NAMELIST;(NEW) Cannot use OR/XOR/AND constructions.
      Did not find commas or $ between or after names
      NAMELIST;(NEW) cannot modify existing variables.

807   Too few valid observations to fit model.
      Moment matrix for regression is singular.
      A perfect fit is obtained for the regression part!
      No variation in LHS variable for regression!

808   ADF(.) needs 3 settings: Variable,Type,#lags.
      ADF, type must = 1 (r.w), 2(drift), or 3(trend).
      Lags for ADF test must be one of 0,1,2,...,10
      Could not compute ADF. See previous diagnostic
      Unstable equation in lagged differences. Sum is >= 1
      Perfect fit in ADF regression equation. EstVar[c]=0.
      Sample has too few observations to carry out test.
      Moment matrix for ADF (regression) test is singular

808   Implied or estimated THETA is zero. Cannot compute ME.
      This is for the Box-Cox model.

809   Fully simultaneous BVP model is not identified

813   Could not evaluate expression in SAMPLE command.'

816   SETPANEL must have both ;GROUP=name and ;PDS=name.
      Did not find variable given in ;GROUP=name.
      ;PDS=name gives an invalid name to use for PDS
      Unable to set up GROUP and PDS variables in SETPANEL
      Using ;PDS=number. NOBS not a multiple of number.
      Group count variable is mismatched to full sample.
      Invalid weighting variable. Nonpositive values were found.
      Number of strata > 5000! Tables overflowing. Must exit.
      Use only one of ;PDS, ;CLUSTER or ;STR in TABLES command.
      Counter <=0. Row = <nnnnnn>, group = <nnnnnn>
      Too many periods to fit two way fixed effects models.
      Noncontiguous panel is too large. Cannot store means.

811   ROWS:  You must give a value from 1 to 1000.

812   Title=... \Name or scalar. Cannot match name after \.
      TITLE=string... String may not exceed 80 characters.

818   DEFAULT: Did not find expected equals sign.'
      Unreadable value given in DEFAULT command.

819   Expected <nnn> specifications in RST/CML list. Found <nnn>.

820   EXEC;:name=list..., name > 7 chars or no = sign found.
      EXEC;:name=list. An unknown name appears in the list.
```

```
821    Quantile Regression needs n <= 100,000.'
       Unable to allocate memory for analysis. Must exit.'

851    Multiple imputation requires M > 1.
       Invalid value given for seed in EXEC command
       ;IMPUTATION=label is not supported for this model command
       Name in ;IMPUTATION=name on command not found in list w/ EXEC
       IMPUTE;LHS=.;RHS=.;Type=Fill$ must have 1 var on L and R.
       Imputation models table is full (30 equations)
```

The following is a generic diagnostic produced when a model command is translated. The general form of the command is

Model ; ... ; Specification... $

Diagnostic 999 occurs when the 'Specification' part of the command, such as Rhs, Lhs, Rh2, and so on, is not recognized.

```
999    The specification ; <XXX spec. is listed.> is not recognized.
```

R28.4 Discrete Choice (CLOGIT) and NLOGIT

Diagnostics with numbers 1,000 and higher are generated by the CLOGIT command or by the command parser or estimators in *NLOGIT*.

```
1000 FIML/NLogit is not enabled in this program.
```

This error occurs when *LIMDEP* encounters a model command such as RPLOGIT that is only enabled in *NLOGIT*.

```
1001   Syntax problem in tree spec or expected ; or $ not found.

1002   Model defines too many alternatives (more than 100).

1003   A choice label appears more than once in the tree specification.

1004   Number of observations not a multiple of # of alternatives.
```

This is expected when you have a fixed choice set.

```
1005   Problem reading labels, or weights for choice based sample.

1006   Number of weights given does not match number of alternatives.

1007   A choice based sampling weight given is not between zero and one.

1008   The choice based sampling weights given do not sum to one.

1009   Expected [ in limb specification was not found.

1010   Expected ( in branch specification was not found.
```

1011 `A branch label appears more than once in the tree.`

1012 `A choice label in a branch spec. is not in ;CHOICES list.`

1013 `Branch specifications are not separated by commas.`

1014 `One or more ;CHOICE labels does not appear in the tree.`

1015 `One or more ;CHOICE labels appears more than once in tree.`

1016 `The model must have either 1 or 3 LHS variables. Check spec.`

1017 `Nested logit model must include ;MODEL:... or ;RHS spec.`
`Found neither Model: nor RhS/Rh2.`
Your model specification is incomplete.

1018 `There is an unidentified variable name in the equation.`
In the ; Model: U (...) part of the command, one of your specified utility functions contains a variable name that is not in your data set.

1019 `Model specification exceeds an internal limit. See documentation.`
`RANK data can only be used for 1 level (nonnested) models.`
You have specified a nested logit model and requested rank data for the observed outcomes. The nested logit model cannot be estimated with ranks data.

1020 `Not used specifically.`
`May show up with a self explanatory message.`

1021 `Using Box-Cox function on a variable that equals 0?`

1022 `Insufficient valid observations to fit a model.`

1023 `Mismatch between current and last models.`
This occurs when you are using the ; Simulation = ... part of *NLOGIT*.

1024 `Failure estimating DISCRETE CHOICE model.`
Since this occurs during an attempt to compute the starting values for other models, if it fails here, it won't succeed in the more complicated model.

1025 `Failed to fit model. See earlier diagnostic.`

1026 `Singular VC may mean model is unidentified. Check tree.`
What looks like convergence of a nested logit model may actually be an unidentified model. In this case, the covariance matrix will show up with at least one column of zeros.

1027 `Models - estimated variance matrix of estimates is singular.`
`Non P.D. 2nd derivatives. Trying BHHH estimator instead.`
This is just a notice. In almost all cases, the Hessian for a model that is not the simple MNL model will fail to be positive definite at the starting values. This does not indicate any kind of problem.

1028 `In ;SIMULATION=list of alts, a name is unknown.`

1029 `Did not find closing ] in labels[list].`

1030 `Error in specification of list in ;Choices=...labels[list].`

1031 `List in ;Choices=...labels[list] must be 1 or NALT values.`

1032
```
Merging SP and RP data. Not possible with 1 line data setup.
Merging SP and RP data requires LHS=choice,NALTi,ALTij form.
Check :MERGERPSP(id=variable, type=variable) for an error.
```

1033
```
Indiv. <nnnnnn> with ID= <nnnnn> has same ID as another
individual.
```
This makes it impossible to merge the data sets.

1034 `Specification error. Scenario must begin with a colon.`

1035 `Expected to find Scenario: specification = value.`

1036 `Unbalanced parentheses in scenario specified.`

1037 `Choice given in scenario: attr(choice...) is not in the model.`

1038 `Cannot identify attribute specified in scenario.`

1039 `Value after = in scenario spec is > 20 characters.`

1040 `Cannot identify RHS value in scenario spec.`

1041 `Transformation asks for divide by zero.`

1042 `Can only analyze 5 scenarios at a time.`

1043 `Did not find any valid observations for simulation.`

1044 `Expected to find ; LIST : name_x ( choices ). Not found.`

1045 `Did not find matching ( or [ in <scenario specification is given>.`

1046
```
Cannot recognize the name  <AAAAAAAA> in <scenario
(specification is given)>.
```

1047 `Same as 1046.`

1048 `None of the attributes requested appear in the model.`

1049 `Model has no free parameters among slopes!`
This occurs during an attempt to fit the MNL model to obtain starting values for a nested logit or some other model.

```
1050  DISC with RANKS. Obs= <nnnnnn>. Alt= <nn>. Bad rank given = <nnnn>.
      DISC w/ RANKS. Incomplete set of ranks given for obs.<nnnnnn>.
```
These are data problems with the coding of the Lhs variable.

```
1051  Singular VC matrix trying to fit MNL model.
```
When the MNL breaks down, it will be impossible to fit a more elaborate model such as a nested logit model.

```
1052  You did not provide ;FCN=label(distn),... for RPL model.

1053  Scaling option is not available with HEV, RPL, or MNP model.
      Ranks data may not be used with HEV, RPL, or MNP model.
      Nested models are not available with HEV, RPL, or MNP model.
      Cannot keep cond. probs. or IVs with HEV, RPL, or MNP model.
      Choice based sampling not useable in HEV, RPL, or MNP model.
```

These diagnostics are produced by problems setting up the scaling option for mixed data sets.

```
1054  Scaling option is not available with one line data setup.
      Ranks data may not be used with one line data setup.
      Choice set may not be variable with one line data setup.
      One line data setup requires ;RHS and/or ;RH2 spec.
      Nested models are not available with one line data setup.
      Cannot keep probabilities or IVs with one line data setup.

1055  Did not find closing paren in ;SCALE(list) spec.
      The list of variables to be scaled has an error.
      Only 40 or fewer variables may be scaled.
      You are attempting to scale the LHS variable.
      The list of values given for SCALE grid is bad.
      Grid must = Lo,Hi,N or Lo,Hi,N,N2. Check spec.
      Grid must have Low > 0 and High > low. Check #s.
      Number of grid points must be 2,3,... up to 20.

1056  Unidentified name in IIA list. Procedure omitted.

1057  More than 5 names in IIA list. Limit is 5.

1058  Size variables only available with (Nested) MNL.

1059  Cannot locate size variable specified.

1060  Model is too large: Number of betas up to 90.
      Model is too large: Number of alphas up to 30.
      Model is too large: Number of gammas up to 15.
      Model is too large: Number of thetas up to 10.

1061  Number of RHS variables is not a multiple of # of choices.
```
This occurs when you are using a one line setup for your data.

```
1062  Expected ;FIX=name[...]. Did not find [ or ].
```

1063 In ;FIX=name[...], name does not exist: <name is given>.

1064 Error in fixed parameter given for <name is given>.

1065 Wrong number of start values given.
This occurs with nested logit and other models, not the random parameters logit model.

Expected ;KERNEL=(...),... Missing parenthesis
The limit on latent effects in KERNEL is 10.
Willingness to pay computations
Error in specification of WTP=name1/name2
Can only save up to 5 WTP values per run.
These diagnostics relate to the kernel model setup.

1066 Command has both ;RHS and Model: U(alts). Inconsistent.

1067 Syntax problem in ;USET:(names list)= list of values.

1068 ;USET: list of parms contains an unrecognized name.

1069 Warning, ;IUSET: # values not equal to # names.

1070 Warning, ;IUSET: # values not equal to # names.

1071 Spec for RPL model is label(type) or [type]. Type=N,C,or L.

1072 Expected ,;$ in COR/SDV/HFN/REM/AR1=list not found.

1073 Invalid value given for correl. or std.dev. in list.

1074 ;COR/SDV=list did not give enough values for matrix.

1075 Error. Expected [in ;EQC=list[value] not found.
Error:Value in EQC=list[value] is not a correlation.
Error. Unrecognized alt name in ;EQC=list[value].
Error:List needs more than 1 name in EQC=list[value].
Error. A name is repeated in ;EQC=list[value].

1076 Your model forces a free parameter equal to a fixed one.

1077 Covariance heterogeneity model needs nonconstant variables.
Invalid parameter name (;label) ',a8,' is a ',a8,'.')

1078 Covariance heterogeneity model not available with HEV model.
Covariance heterogeneity model is only for 2 level models.
Covariance heterogeneity model needs 2 or more branches.

1079 At least one variance in the HEV model must be fixed.
In *NLOGIT*, in the heteroscedastic extreme value, you have specified the model so that all the variances are free. But, for identification, one of them must be fixed.

1080 Multiple observation RPL/MNP data must be individual.

1081 `Mismatch of # indivs. and number implied by groups.`
`WARNING   Halton method is limited to 25 random parameters.`

1082 `Not used.`

1083 `MODEL followed by a colon was expected, not found.`

1084 `Expected equation specs. of form U(...) after MODEL.`

1085 `Unidentified name found in <string is given>.`
This occurs during translation of ; Model: U (...) specifications.

1086 `U(list) must define only choices,branches, or limbs.`

1087 `An equals sign was not found where expected in utility function definition.`

1088 `Mismatched [ or ( in parameter value specification.`

1089 `Could not interpret string; expected to find number.`

1090 `Expected to find ;IVSET:=defn. at this point.`

1091 `Expected to find a list of names in parens in IVSET.`

1092 `IVSET:( list ) ... Unidentified name appears in (list).`

1093 `You have given a spec for an IV parm that is fixed at 1.`

1094 `You have specified an IV parameter more than once.`

1095 `Count variable  <nnnnnn> at row <nnnnnn> equals <nnnn>.`
The peculiar value for the count variable has thrown off the counter that keeps track of where the estimator is in the data set.

1096 `Choice variable  <AAAAAAAA>  at row  <nnnnn>: Choice= <nnnnn>.`
The most likely cause is a coding error. Check for bad data.

1097 `Obs. <nnnnnn>: Choice set contains <nnnn> <nnnn> times.`
The choice variable for individual data has more than one 1.0 in it. *LIMDEP* cannot determine which alternative is chosen.

1098 `Obs. <nnnnnn> alt. <nnn> is not an integer nor a proportion.`

1099 `Obs. <nnnnnn> responses should sum to 1.0. Sum is <xxxxxx>.`

1100 `Cannot classify obs. <nnnnnn> as IND, PROPs, or  FREQs.`
Your data appear to be a mix of individual and frequency data. This occurs when an individual’s Lhs variable data include zeros. It then becomes difficult to determine what kind of data you have. You can settle the question by including ; Frequencies in your command, if that is appropriate.

```
1101  # of parms in < list > greater than # choices in U(list).

1102  RANK data can only be used for 1 level (nonnested) model.

1103  Wrong number of variables given in ;CLASSP=list.
      ;CLASSP=list contains ONE. Cannot save P(j|i) in ONE.

1104  Negative value in NLRP;Tau=value is ignored
      Negative value in GMXL;Tau=value is ignored
      Value not in [0,1] in GMXL;Gamma=value
      Unknown name in ;RPASC=list. Spec. ignored.'
```

Diagnostics 1121-1144 are produced by the nonlinear random parameters logit model (NLRPLOGIT).

```
1121  Too many parameters in list (over 150)

1122  num_symbol, num negative or greater than 150

1123  No. of start values must equal no. of labels.

1124  NLRPLogit requires ;Start=starting values.

1125  Error reading starting values for NLRPLogit

1126  Error in ;FIX=list of labels for NLRPLogit.

1127  Invalid parameter name (;label) ',a8,' is a ',a8,'.'

1128  Fn. name conflicts with var. or other name.

1129  Unbalanced parentheses in function defn.

1130  Table overflow. Function is too complex.

1131  Error in function. See earlier error msg.

1132  Expected to find ;Model:U(...) = name / ...

1133  Utility spec uses a function not in the table

1134  ;Fnj=function name=function defnn.

1135  Alternative function name may not equal a label

1136  Expected ending ] in name[...] was not found

1137  Unknown name appears in list in name[list]
1138  WTP setup for NLRP must be alt[xvar/xvar].

1139  Alt name in WTP spec for NLRP is unknown

1140  X var name in Alt[Xvar/Yvar] is unknown.
```

```
1141  Y var name in Alt[Xvar/Yvar] is unknown.

1142  Expected ;888:(xname,blabel) colon not found

1143  Expected (xname,bname) found incorrect specs.'

1144  Table full,25 specs for 888:(xname,bname)/...

1151  User fn. in RPMIN/MAX is nonpositive. Using Log(.)?

1152  Numerical underflow Product of F(i,r,t) is too small.

1153  Numerical overflow Product of F(i,r,t) is too large.
```

LIMDEP 11 References

Abowd, J. and H. Farber, H. (1982) ‘Job Queues and the Union Status of Workers,’ *Industrial and Labor Relations Review*, 35, pp. 354-367.

Abramovitz, M. and Stegun, I. (1972) *Handbook of Mathematical Functions*, Dover Press, New York.

Ackerberg, D. and Devereux, P. (2009) ‘Improved Jive Estimators for Overidentified Linear Models with and without Heteroskedasticity,’ *Review of Economics and Statistics*, 91, 2, pp. 351-362.

Agresti, A. (1984) *Analysis of Ordinal Categorical Data*, John Wiley and Sons, New York.

Ahn, S. and Schmidt, P. (1995) ‘Efficient Estimation of Models for Dynamic Panel Data,’ *Journal of Econometrics*, 68, pp. 3-38.

Ai, C. and Norton, E. (2003) ‘Interaction Terms in Logit and Probit Models,’ *Economics Letters*, Elsevier, 80, 1, pp. 123-129.

Aigner, D., Lovell, K., and Schmidt, P. (1977) ‘Formulation and Estimation of Stochastic Frontier Production Function Models,’ *Journal of Econometrics*, 6, pp. 21-37.

Alvarez, A., Amsler, C., Orea, L., and Schmidt, P. (2006) ‘Interpreting and Testing the Scaling Property in Models where Inefficiency Depends on Firm Characteristics,’ *Journal of Productivity Analysis*, 25, 3, pp. 201-212.

Alvarez, A., Arias, C., and Greene, W. (2006) ‘Fixed Management and Time Invariant Efficiency in a Random Coefficients Model,’ Manuscript, Department of Economics, University of Oviedo, Oviedo.

Amemiya, T. (1973) ‘Regression Analysis When the Variance of the Dependent Variable is Proportional to the Square of Its Expectation,’ *Journal of the American Statistical Association*, 68, pp. 928-934.

Amemiya, T. (1981) ‘Qualitative Response Models: A Survey,’ *Journal of Economic Literature*, 19, pp. 1483-1536.

Amemiya, T. (1984) ‘Censored or Truncated Regression Models, Symposium,’ *Journal of Econometrics*, 24, 1/2, pp. 1-222.

Amemiya, T. (1987) *Advanced Econometrics*, Harvard University Press, Cambridge.

Amemiya, T. and MaCurdy, T. (1986) ‘Instrumental Variables Estimation of an Error Components Model,’ *Econometrica*, 54, pp. 869-880.

Anderson, S. and Newell, R. (2003) ‘Simplified Marginal Effects in Discrete Choice Models,’ *Economics Letters*, 81, 3, pp. 321-326.

Angrist, J.D., Imbens, G.W., and Krueger, A.B. (1999) ‘Jackknife Instrumental Variables Estimation,’ *Journal of Applied Econometrics*, 14, 1, pp. 57-67.

Angrist, J.D. and Pischke, J. (2009) *Mostly Harmless Econometrics: An Empiricist’s Companion*, Princeton University Press, Princeton.

Antweiler, W. (2001) ‘Nested Random Effects Estimation in Unbalanced Panel Data,’ *Journal of Econometrics*, 101, pp. 295-313.

Arellano, M. and Bond, S. (1991) ‘Some Tests of Specification for Panel Data: Monte Carlo Evidence and an Application to Employment Equations’ *Review of Economic Studies*, 58, pp. 277-297.

Arellano, M. and Bond, S. (1998) ‘Dynamic Panel Data Estimation Using DPD98 for Gauss: A Guide for Users,’ CEMFI, Madrid.

Arellano, M. and Bover, O. (1995) ‘Another Look at the Instrumental Variable Estimation of Error-Components Models,’ *Journal of Econometrics*, 68, pp. 29-51.

Atkinson, S. and Cornwell, C. (1994) 'Parametric Estimation of Technical and Allocative Inefficiency with Panel Data,' *International Economic Review*, 35, 1, pp. 231-244.

Baltagi, B. (2005) *Econometric Analysis of Panel Data*, 3rd Edition, John Wiley and Sons, New York.

Baltagi, B. and Li, Q. (1990) 'A Comparison of Variance Components Estimators Using Balanced Versus Unbalanced Data,' *Econometric Theory*, 6, 2, pp. 283-285.

Barnow, B., Cain, G., and Goldberger, A. (1981) 'Issues in the Analysis of Selection Bias,' Department of Economics, University of Wisconsin, Madison.

Battese, G. and Coelli, T. (1988) 'Prediction of Firm-Level Technical Efficiencies with a Generalized Frontier Production Function and Panel Data,' *Journal of Econometrics*, 38, pp. 387-399.

Battese, G. and Coelli, T. (1992) 'Frontier Production Functions, Technical Efficiency and Panel Data: With Application to Paddy Farmers in India,' *Journal of Productivity Analysis*, 3, pp. 153-169.

Battese, G. and Coelli, T. (1995) 'A Model for Technical Inefficiency Effects in a Stochastic Frontier Production Function for Panel Data,' *Empirical Economics*, 20, pp. 325-332.

Battese, G. and Coelli, T. (eds.) (1997) 'Efficiency and Productivity Measurement,' *Journal of Productivity Analysis*, 8 (entire issue).

Battese, G. and Corra, G. (1977) 'Estimation of a Production Frontier Model: With Application for the Pastoral Zone of Eastern Australia,' *Australian Journal of Agricultural Economics*, 21, pp. 167-179.

Bauer, P. (1990) 'A Survey of Recent Econometric Developments in Frontier Estimation,' *Journal of Econometrics*, 46, pp. 21-39.

Beach, C. and MacKinnon, J. (1978) 'A Maximum Likelihood Procedure for Regression with Autocorrelated Errors,' *Econometrica*, 46, pp. 51-58.

Beck, N. and Katz, J. (1995) 'What To Do (And Not To Do) with Time Series-Cross Section Data in Comparative Politics,' *American Political Science Review*, 89, pp. 634-647.

Becker, S. and Ichino, A. (2002) 'Estimation of Average Treatment Effects Based on Propensity Scores,' *The Stata Journal*, 2, pp. 358-377.

Beckers, D. and Hammond, C. (1987) 'A Tractable Log-Likelihood Function for the Normal-Gamma Stochastic Frontier Model,' *Economics Letters*, 24, pp. 33-38.

Beggs, J., Cardell, S., and Hausman, J. (1981) 'Assessing the Potential Demand for Electric Cars,' *Journal of Econometrics*, 17, pp. 1-19.

Belotti, F. and Ilardi, G. (2014) 'Consistent Inference in Fixed-Effects Stochastic Frontier Models,' Working Paper (November 7 R&R Version), University of Rome, Italy.

Belsley, D. (1980) 'On the Efficient Computation of the Nonlinear Full-Information Maximum Likelihood Estimator,' *Journal of Econometrics*, 14, pp. 203-224.

Belsley, D., Kuh, E., and Welsh, R. (1980) *Regression Diagnostics*, John Wiley and Sons, New York.

Bera, A., Jarque, C., and Lee, L. (1984) 'Testing the Normality Assumption in Limited Dependent Variable Models,' *International Economic Review*, 25, pp. 563-578.

Bera, A. and Sharma, S. (1999) 'Estimating Production Uncertainty in Stochastic Frontier Production Function Models.' *Journal of Productivity Analysis*, 12, pp. 187-210.

Berndt, E. (1991) *The Practice of Econometrics,* Addison Wesley, New York.

Berndt, E., Hall, B., Hall, R., and Hausman, J. (1974) 'Estimation and Inference in Nonlinear Structural Models,' *Annals of Economic and Social Measurement*, 3/4, pp. 653-666.

Berndt, E. and Wood, D. (1975) 'Technology, Prices, and the Derived Demand for Energy,' *Review of Economics and Statistics*, 57, pp. 376-384.

Berry, S., Levinsohn, J. and Pakes, A. (1995) 'Automobile Prices in Market Equilibrium,' *Econometrica*, 63, pp. 841-890.

Bhargava, A. and Sargan, J. (1983) 'Estimating Dynamic Random Effects Models from Panel Data Covering Short Periods,' *Econometrica*, 51, pp. 221-236.

Bhat, C. (1994) 'Imputing a Continuous Income Variable from Grouped and Missing Income Observations,' *Economics Letters*, 46, 4, pp. 311-320.

Bhat, C. (1999) 'Quasi-Random Maximum Simulated Likelihood Estimation of the Mixed Logit Model,' Msp., Department of Civil Engineering, University of Texas, Austin.

Bhat, C. (2001) 'Quasi-Random Maximum Simulated Likelihood Estimation of the Mixed Multinomial Logit Model,' *Transportation Research*, 35B, pp. 677-693.

Bhat, C. and Castelar, S. (2000) 'A Unified Mixed Logit Framework for Modeling Revealed and Stated Preferences: Formulation and Application to Congestion Pricing Analysis in the San Francisco Bay Area,' Manuscript, Department of Civil Engineering, University of Texas, Austin.

Bloom, D. and Killingsworth, M. (1985) 'Correcting for Truncation Bias Caused by a Latent Truncation Variable,' *Journal of Econometrics*, 27, 1, pp. 131-135.

Blundell, R., Griffith, R., and Windmeijer, F. (2002) 'Individual Effects and Dynamics in Count Data Models,' *Journal of Econometrics*, 108, pp. 113-131.

Blundell, R. and Smith, R. (1986) 'An Exogeneity Test for a Simultaneous Equation Tobit Model with an Application to Labor Supply,' *Econometrica*, 54, 3, pp. 679-685.

Bollerslev, T. (1986) 'Generalized Autoregressive Conditional Heteroscedasticity,' *Journal of Econometrics*, 31, pp. 307-327.

Bollerslev, T. and Ghysels, E. (1996) 'Periodic Autoregressive Conditional Heteroscedasticity,' *Journal of Business and Economic Statistics*, 14, pp. 139-151.

Bollerslev, T. and Wooldridge, J. (1992) 'Quasi Maximum Likelihood Estimation and Inference in Dynamic Models with Time Varying Covariances,' *Economic Reviews*, 11, pp. 143-172.

Bowman, K. and Shenton, L. (1975) 'Omnibus Contours for Departures from Normality Based on $\surd b_1$ and b_2,' *Biometrika* 62, 2, pp. 243-250.

Box, G. and Jenkins, G. (1984) *Time Series Analysis: Forecasting and Control*, 2nd Edition, Holden Day, New York.

Boyes, W., Hoffman, D., and Low, S. (1989) 'An Econometric Analysis of the Bank Credit Scoring Problem,' *Journal of Econometrics*, 40, pp. 3-14.

Breusch, T. and Pagan, A. (1979) 'A Simple Test for Heteroscedasticity and Random Coefficient Variation,' *Econometrica*, 47, pp. 1287-1294.

Breusch, T. and Pagan, A. (1980) 'The LM Test and its Application to Model Specification in Econometrics,' *Review of Economic Studies*, 47, pp. 239-254.

Brown, R., Durbin, J., and Evans, J. (1972) 'Techniques for Testing the Constancy of Regression Relationships Over Time,' *Journal of the Royal Statistical Society*, Series B, 37, pp. 149-172.

Brown, S., Greene, W., Harris, M., and Taylor, K. (2015) 'An Inverse Hyperbolic Sine Heteroscedastic Latent Class Panel Tobit Model: An Application to Modeling Charitable Donations,' *Economic Modelling*, 50, pp. 228-236

Brownstone, D. and Train, K. (1999) 'Forecasting New Product Penetration with Flexible Substitution Patterns,' *Journal of Econometrics*, 89, pp. 109-129.

Burbidge, J.B., Magee, L., and Robb, A.L. (1988) 'Alternative Transformations to Handle Extreme Values of the Dependent Variable,' *Journal of the American Statistical Association*, 83, pp. 123-127.

Burnett, N. (1997) 'Gender Economics Courses in Liberal Arts Colleges,' *Journal of Economic Education*, 28, 4, pp. 369-377.

Butler, J. and Chatterjee, P. (1997) ‘Tests of the Specification of Univariate and Bivariate Ordered Probit,’ *Review of Economics and Statistics*, 79, 2, pp. 343-347.

Butler, J. and Moffitt, R. (1982) ‘A Computationally Efficient Quadrature Procedure for the One Factor Multinomial Probit Model,’ *Econometrica*, 50, 3, pp. 761-764.

Cameron, C. and Trivedi, P. (1986) ‘Econometric Models Based on Count Data: Comparisons and Applications of Some Estimators,’ *Journal of Applied Econometrics*, 1, 1, pp. 29-54.

Cameron, C. and Trivedi, P. (1990) ‘Regression Based Tests for Overdispersion in the Poisson Regression Model,’ *Journal of Econometrics*, 46, pp. 347-364.

Cameron, C. and Trivedi, P. (1998) *Regression Analysis of Count Data*, Cambridge University Press, New York.

Cameron, C. and Trivedi, P. (2005) *Microeconometrics: Methods and Applications*, Cambridge University Press, New York.

Cecchetti, S. (1986) ‘The Frequency of Price Adjustment: A Study of the Newsstand Prices of Magazines,’ *Journal of Econometrics*, 31, pp. 255-274.

Chamberlain, G. (1980) ‘Analysis of Covariance with Qualitative Data,’ *Review of Economic Studies*, 47, pp. 225-238.

Chatfield, C. (1996) *The Analysis of Time Series: An Introduction*, 5th Edition, Chapman and Hall, London.

Chatterjee, S. and Price, B. (1991) *Regression Analysis by Example*, 2nd Edition, John Wiley and Sons, New York.

Chen, Y., Schmidt, P., and Wang, H. (2014), ‘Consistent Estimation of the Fixed-Effects Stochastic Frontier Model,’ *Journal of Econometrics*, 181, 2, pp. 65–76.

Chesher, A. and Irish, M. (1987) ‘Residual Analysis in the Grouped Data and Censored Normal Linear Model,’ *Journal of Econometrics*, 34, pp. 33-62.

Cheung, C. and Goldberger, A. (1984) ‘Proportional Projections in Limited Dependent Variable Models,’ *Econometrica*, 52, pp. 531-534.

Christensen, L. and Greene, W. (1976) ‘Economies of Scale in U.S. Electric Power Generation,’ *Journal of Political Economy*, 84, pp. 625-656.

Christofides, L., Stengos, T., and Swidinsky, R. (1997) ‘On the Calculation of Marginal Effects in the Bivariate Probit Model,’ *Economics Letters*, 54, 3, pp. 203-208.

Cleveland, W. (1979) ‘Robust Locally Weighted Regression and Smoothing Scatterplots,’ *Journal of the American Statistical Association*, 74, 368, pp. 829-836.

Cockburn, I., Wang, P., and Puterman, B. (1998) ‘Analysis of Patent Data - A Mixed Poisson Regression Model Approach,’ *Journal of Business and Economic Statistics*, 16, 1, pp. 27-41.

Coelli, T. (1996a) ‘A Guide to DEAP Version 2.1: A Data Envelopment Analysis (Computer Program),’ CEPA Working Paper 96/08, Department of Econometrics, University of New England, Armidale.

Coelli, T. (1996b) ‘A Guide to FRONTIER Version 4.1: A Computer Program for Stochastic Frontier Production and Cost Function Estimation,’ CEPA Working Paper 96/07, Department of Econometrics, University of New England, Armidale.

Coelli, T., Prasada Rao, D., O’Donnell, C., and Battese, G. (2005), *An Introduction to Efficiency and Productivity Analysis*, 2nd Edition, Springer, New York.

Colombi, R. (2010), ‘A Skew Normal Stochastic Frontier Model for Panel Data,’ Proceedings of the 45th Scientific Meeting of the Italian Statistical Society.

Colombi, R., Martini, G., and Vittadini, G. (2011) ‘A Stochastic Frontier Model with Short-Run and Long-Run Inefficiency Random Effects,’ Working Paper, Department of Economics and Technology Management, University of Bergamo, Italy.

Consul, P. and Jain, G. (1973) ‘A Generalization of the Poisson Distribution,’ *Technometrics*, 15, pp. 791-799

Cornwell, C. and Rupert, P. (1988) ‘Efficient Estimation with Panel Data: An Empirical Comparison of Instrumental Variable Estimators,’ *Journal of Applied Econometrics*, 3, pp. 149-155.

Cornwell, C., Schmidt, P., and Sickles, R. (1990) ‘Production Frontiers with Cross Sectional and Time Series Variation in Efficiency Levels,’ *Journal of Econometrics*, 46, pp. 185-200.

Cotton, J. (1988) ‘On the Decomposition of Wage Differentials,’ *Review of Economics and Statistics*, 70, 2, 1988, pp. 236-239.

Cox, D. (1961) ‘Tests of Separate Families of Hypotheses,’ Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press, Berkeley.

Cox, D. (1972) ‘Regression Models and Life Tables,’ *Journal of the Royal Statistical Society*, Series B, 34, pp. 187-220.

Cox, D. (1975) ‘Partial Likelihood,’ *Biometrika*, 62, pp. 269-276.

Cox, D. and Oakes, R. (1984) *Analysis of Survival Data*, Chapman and Hall, London.

Cragg, J. (1971) ‘Some Statistical Models for Limited Dependent Variables with Application to the Demand for Durable Goods,’ *Econometrica*, 39, pp. 829-844.

Creel, M. and Loomis, J. (1990) ‘Theoretical and Empirical Advantages of Truncated Count Data Estimators for Analysis of Deer Hunting in California,’ *American Journal of Agricultural Economics*, 72, pp. 434-441.

Crepon, B. and Duguet, E. (1995) ‘Research and Development, Competition and Innovation: Pseudo Maximum Likelihood and Simulated Maximum Likelihood Applied to Count Data Models with Heterogeneity,’ *Journal of Econometrics*, 79, 2, pp. 355-378.

Cutler, S. and Ederer, F. (1958) ‘Maximum Utilization of the Life Table in Analyzing Survival,’ *Journal of Chronic Disorders*, pp. 699-712.

Davidson, R. and MacKinnon, J.G. (1981) ‘Several Tests for Model Specification in the Presence of Multiple Alternatives,’ *Econometrica*, 49, pp. 781-793.

Davidson, R. and MacKinnon, J.G. (1993) *Estimation and Inference in Econometrics*, Oxford University Press, Oxford.

Davidson, R., and MacKinnon, J.G. (2004) *Econometric Theory and Methods*, Oxford University Press, New York.

Davidson, R., and MacKinnon, J.G. (2006) ‘The Case Against JIVE,’ *Journal of Applied Econometrics*, 6, pp. 827-833.

Daymont, T. and Andrisani, P. (1984) ‘Job Preferences, College Major and the Gender Gap in Earnings,’ *Journal of Human Resources* 19, 3, pp. 408-428.

Deheija, R. and Wahba, S. (1999) ‘Causal Effects in Nonexperimental Studies: Reevaluation of the Evaluation of Training Programs,’ *Journal of the American Statistical Association*, 94, pp. 1052-1062.

Dehejia, R. and Wahba, S. (2002) ‘Propensity Score Matching Methods for Nonexperimental Causal Studies,’ *Review of Economics and Statistics*, 84, 1, pp. 151-161.

DeMaris, A. (2004) *Regression with Social Data: Modeling Continuous and Limited Response Variables*, John Wiley and Sons, New York.

Dickey, D. and Fuller, W. (1979) ‘Distribution of the Estimators for Autoregressive Time Series with a Unit Root,’ *Journal of the American Statistical Association*, 74, pp. 427-431.

Diggle, P., Liang, K., and Zeger, S. (1994) *Analysis of Longitudinal Data*, Clarendon Press, Oxford.

Efron, B. (1979) ‘Bootstrapping Methods: Another Look at the Jackknife,’ *Annals of Statistics*, 7, pp. 1-26.

Efron, B. (1998) *An Introduction to the Bootstrap*, John Wiley and Sons, New York.

Efron, B. and Tibshirani, R. (1986) ‘Bootstrap Measures for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy,’ *Statistical Science*, 1, pp. 54-77.

Enders, W. (2003) *Applied Econometric Time Series*, 2nd Edition, John Wiley and Sons, New York.

Engle, R. (1982) ‘Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflations,’ *Econometrica*, 50, pp. 987-1008.

Engle, R., Lilien, D., and Robins, R. (1987) ‘Estimating Time Varying Risk Premia in the Term Structure: The ARCH-M Model,’ *Econometrica*, 55, pp. 391-407.

Englin, J. and Shonkwiler, J. (1995) ‘Estimating Social Welfare Using Count Data Models: An Application to Long-Run Recreation Demand Under Conditions of Endogenous Stratification and Truncation,’ *Review of Economics and Statistics*, 77, pp. 104-112.

Estrella, A. (1998) ‘A New Measure of Fit for Equations with Dichotomous Dependent Variables,’ *Journal of Business and Economic Statistics*, 16, 2, pp. 198-205.

Fair, R. (1977) ‘A Note on Computation of the Tobit Estimator,’ *Econometrica*, 45, pp. 1723-1727.

Fair, R. (1978) ‘A Theory of Extramarital Affairs,’ *Journal of Political Economy*, 86, pp. 45-61.

Fare, R., Grosskopf, S., and Lovell, C.A.K. (1994) *Production Frontiers*, Cambridge University Press, Cambridge.

Filippini, M. and Greene, W. (2016) ‘Persistent and Transient Productive Inefficiency: A Maximum Simulated Likelihood Approach,’ *Journal of Productivity Analysis*, 45, forthcoming.

Fin, T. and Schmidt, P. (1984) ‘A Test of the Tobit Specification Against an Alternative Suggested by Cragg,’ *Review of Economics and Statistics*, 66, pp. 174-177.

Fiorentini, G., Calzolari, G., and Panattoni, L. (1996) ‘Analytic Derivatives and the Computation of GARCH Estimates,’ *Journal of Applied Econometrics*, 11, pp. 399-417.

Fletcher, R. (1980) *Practical Methods of Optimization*, John Wiley and Sons, New York.

Fomby, T., Hill, R., and Johnson, S. (1984) *Advanced Econometric Methods*, Springer Verlag, Heidelberg.

Fry, T. (1991) ‘A Generalized Logistic Tobit Model,’ Mimeo, Department of Econometrics, Monash University, Clayton.

Glass, G. and Hopkins, K. (1996) *Statistical Methods in Education and Psychology*, 3rd Edition, Allyn and Bacon, Needham Heights.

Glejser, H. (1965) ‘A New Test for Heteroskedasticity,’ *Journal of the American Statistical Association*, 60, pp. 539-547.

Glewwe, P. (1997) ‘A Test of the Normality Assumption in the Ordered Probit Model,’ *Econometric Reviews*, 16, pp. 1-19.

Godfrey, L. G. (1978) ‘Testing Against General Autoregressive and Moving Average Error Models when the Regressors Include Lagged Dependent Variables,’ *Econometrica*, 46, pp. 1293-1302.

Goldberg, P. (1995) ‘Product Differentiation and Oligopoly in International Markets: The Case of the U.S. Automobile Industry,’ *Econometrica*, 63, pp. 891-952.

Goldfeld, S. and Quandt, R. (1972) *Nonlinear Methods in Econometrics*, North-Holland, Amsterdam.

Gong, X., van Soest, A., and Villagomez, E. (2000) ‘Mobility in the Urban Labor Market: A Panel Data Analysis for Mexico,’ IZA Working Paper 213, Bonn.

Gourieroux, C., Monfort, A., and Trognon, A. (1984) ‘Pseudo Maximum Likelihood Methods: Applications to Poisson Models,’ *Econometrica*, 52, pp. 701-720.

Greene, W. (1981) ‘Sample Selection Bias as a Specification Error: Comment,’ *Econometrica*, 49, pp. 795-798.

Greene, W. (1983) ‘Estimation of Limited Dependent Variable Models by Ordinary Least Squares and the Method of Moments,’ *Journal of Econometrics*, 21, pp. 195-212.

Greene, W. (1984) ‘Estimation of the Correlation Coefficient in a Bivariate Probit Model Using the Method of Moments,’ *Economics Letters*, 16, pp. 285-291.

Greene, W. (1990a) ‘A Gamma Distributed Stochastic Frontier Model,’ *Journal of Econometrics*, 46, pp. 141-163.

Greene, W. (1990b) ‘Multiple Roots of the Tobit Log-Likelihood,’ *Journal of Econometrics*, 46, pp. 365-380.

Greene, W. (1991) ‘The Econometric Approach to Efficiency Measurement,’ Working Paper 93-20, Department of Economics, Stern School of Business, New York University, New York.

Greene, W. (1992) ‘A Statistical Model for Credit Scoring,’ Working Paper EC-92-29, Department of Economics, Stern School of Business, New York University, New York.

Greene, W. (1994) ‘Accounting for Excess Zeros and Sample Selection in Poisson and Negative Binomial Regression Models,’ Working Paper 94-10, Department of Economics, Stern School of Business, New York University, New York.

Greene, W. (1996) ‘Models for Count Data,’ Survey Paper, Manuscript, Department of Economics, Stern School of Business, New York University, New York.

Greene, W. (1997a) ‘Frontier Production Functions,’ in Pesaran, M. and Schmidt, P. (eds.), *Handbook of Applied Econometrics: Microeconometrics*, Oxford University Press, Oxford.

Greene, W. (1998) ‘Gender Economics Courses in Liberal Arts Colleges: Further Results,’ *Journal of Economic Education*, 29, 4, pp. 291-300.

Greene, W. (1999) ‘Marginal Effects in the Censored Regression Model,’ *Economics Letters*, 64, 1, pp. 43-50.

Greene, W. (2000a) ‘Simulated Likelihood Estimation of the Normal-Gamma Stochastic Frontier Model,’ Working Paper 00-05, Department of Economics, Stern School of Business, New York University, New York.

Greene, W. (2001) ‘Fixed and Random Effects in Nonlinear Models,’ Working Paper 01-01, Department of Economics, Stern School of Business, New York University, New York.

Greene, W. (2004a) ‘The Behaviour of the Maximum Likelihood Estimator of Limited Dependent Variable Models in the Presence of Fixed Effects,’ *Econometrics Journal*, 7, pp. 98-119.

Greene. W. (2004b) ‘Fixed Effects and Bias Due to the Incidental Parameters Problem in the Tobit Model,’ *Econometric Reviews*, 23, 2, pp. 125-148.

Greene, W. (2006a) ‘A General Approach to Incorporating Selectivity in a Model,’ Working Paper 06-10, Department of Economics, Stern School of Business, New York University, New York.

Greene, W. (2006b) ‘Econometric Analysis of Panel Data,’ http://pages.stern.nyu.edu/~wgreene/Econometrics/PanelDataNotes-6.ppt.

Greene, W. (2007) ‘Functional Form and Heterogeneity in Models for Count Data,’ *Foundations and Trends in Econometrics*, 1, 2, pp. 113-218.

Greene, W. (2008a) ‘Functional Forms for the Negative Binomial Model for Count Data,’ *Economics Letters*, 99, 3, 2008, pp. 585-590.

Greene, W. (2008b) ‘The Econometric Approach to Efficiency Analysis,’ in Fried, H., Lovell, K., and Schmidt, S. (eds.) *The Measurement of Productive Efficiency and Productivity Growth*, Oxford University Press, Oxford.

Greene, W. (2010) ‘A Sample Selection Corrected Stochastic Frontier Model,’ *Journal of Productivity Analysis*, 34, 1, pp. 15-24.

Greene, W. (2011) ‘Fixed Effects Vector Decomposition: A Magical Solution to the Problem of Time Invariant Variables in Fixed Effects Models?,’ *Political Analysis*, 19, 2, pp. 135-146 and pp. 170-172.

Greene, W. (2012) *Econometric Analysis*, 7th Edition, Prentice Hall, Englewood Cliffs.
Greene, W. and Hensher, D. (2010) *Modeling Ordered Choices*, Cambridge University Press, Cambridge.
Greene, W. and Seaks, T. (1991) 'The Restricted Least Squares Estimator: A Pedagogical Note,' *Review of Economics and Statistics*, 73, pp. 563-567.
Greene, W., Seaks, T., and Greene, L. (1995) 'Estimating the Functional Form of the Independent Variables in Probit Models,' *Applied Economics*, 27, 2, pp. 193-196.
Grogger, J. and Carson, R. (1991) 'Models for Truncated Counts,' *Journal of Applied Econometrics*, 6, pp. 225-238.
Gross, J. and Clark, V. (1975) *Survival Distributions: Reliability Applications in the Biomedical Sciences*, John Wiley and Sons, New York.
Gruvaeus, G. and Joreskog, K. (1970) 'A Computer Program for Minimizing a Function of Several Variables,' *Educational Testing Services, Research Bulletin* 70-14.
Gujarati, D. (1988) *Basic Econometrics*, McGraw-Hill, New York, p. 502.
Gurmu, S. (1991) 'Tests for Detecting Overdispersion in the Positive Poisson Regression Model,' *Journal of Business and Economic Statistics*, 9, pp. 215-222.
Haberman, S. (1980) *Analysis of Qualitative Data*, Academic Press, New York.
Hadri, K. and Whittaker, J. (1999) 'Efficiency, Environmental Contaminants and Farm Size: Testing for Links Using Stochastic Production Frontiers,' *Journal of Applied Economics*, 2, 2, pp. 337-356.
Hadri, K., Guermat, C. and Whittaker, J. (2003a) 'Estimating Farm Efficiency in the Presence of Double Heteroscedasticity Using Panel Data,' *Journal of Applied Economics*, 6, 2, pp. 255-268.
Hadri, K. Guermat, C. and Whittaker, J. (2003b) 'Estimation of Technical Inefficiency Effects Using Panel Data and Doubly Heteroscedastic Stochastic Production Frontiers,' *Empirical Economics*, 28, 1, pp. 203-222.
Hajargasht, G. (2015) 'Stochastic Frontiers with a Rayleigh Distribution,' *Journal of Productivity Analysis*, 44, 2, pp. 199-208.
Hamilton, J. (1994) *Time Series Analysis*, Princeton University Press, Princeton.
Han, A. (1987) 'A Nonparametric Analysis of a Generalized Regression Model: The Maximum Rank Correlation Estimator,' *Journal of Econometrics*, 35, pp. 303-316.
Han, A. and Hausman, J. (1986) 'Semiparametric Estimation of Duration and Competing Risk Models,' Working Paper 450, Department of Economics, MIT, Cambridge.
Han, C., Orea, L., and Schmidt, P. (2005) 'Estimation of a Panel Data Model with Parametric Temporal Variation in Individual Effects,' *Journal of Econometrics*, 126, 2, pp. 241-267.
Hansen, L. (1982) 'Large Sample Properties of Generalized Method of Moments Estimators,' *Econometrica*, 50, pp. 1029-1054.
Hansen, L. and Singleton, K. (1982) 'Generalized Instrumental Variables Estimation of Nonlinear Rational Expectations Models,' *Econometrica*, 50, pp. 1269-1286.
Harris, M. and Zhao, X. (2007) 'A Zero-Inflated Ordered Probit Model, with an Application to Modelling Tobacco Consumption,' *Journal of Econometrics*, 141, 2, pp. 1073-1099.
Harvey, A. (1976) 'Estimating Regression Models with Multiplicative Heteroskedasticity,' *Econometrica*, 44, pp. 461-465.
Harvey, A. (1993) *The Econometric Analysis of Time Series*, 2nd Edition, MIT Press, Cambridge.
Hastings, N., Evans, M., and Peacock, B. (1993) *Statistical Distributions*, 2nd Edition, John Wiley and Sons, New York.

Hatanaka, T. (1974) ‘An Efficient Two-Step Estimator for the Dynamic Adjustment Model with Autocorrelated Errors,’ *Journal of Econometrics*, 2, pp. 199-220.

Hausman, J. (1978) ‘Specification Tests in Econometrics,’ *Econometrica*, 46, pp. 1251-1271.

Hausman, J., Hall, B., and Griliches, Z. (1984) ‘Econometric Models for Count Data with an Application to the Patents - R&D Relationship,’ *Econometrica*, 52, pp. 909-938.

Hausman, J. and McFadden, D. (1984) ‘Specification Tests for the Multinomial Logit Model,’ *Econometrica*, 52, pp. 1219-1240.

Hausman, J. and Taylor, W. (1981) ‘Panel Data and Unobservable Individual Effects,’ *Econometrica*, 49, pp. 1377-1398.

Hausman, J. and Wise, D. (1979) ‘Attrition Bias in Experimental and Panel Data: The Gary Income Maintenance Experiment,’ Econometrica, 47, pp. 455-473.

Hay, D. (1980) ‘Selectivity Bias in a Simultaneous Logit - OLS Model,’ Manuscript, Department of Economics, University of Southern California, Los Angeles.

Hayashi, F. (2000) *Econometrics*, Princeton University Press, Princeton.

He, B., Xie, M., Goh, T., and Tsui, K. (2002) ‘Control Charts Based on Generalized Poisson Model for Count Data,’ The Logistics Institute, Georgia Institute of Technology, Atlanta.

Heckman, J. (1979) ‘Sample Selection Bias as a Specification Error,’ *Econometrica*, 47, pp. 153-161.

Heckman, J. (1981) ‘The Incidental Parameters Problem and the Problem of Initial Conditions in Estimating a Discrete Time-Discrete Data Stochastic Process,’ in Manski, C. and McFadden, D. (eds.), *Structural Analysis of Discrete Data with Econometric Applications*, MIT Press, Cambridge, pp. 114-178.

Heckman, J. and MaCurdy, T. (1980) ‘A Life Cycle Model of Female Labor Supply,’ *Review of Economic Studies*, 47, pp. 247-283.

Heckman, J., Ichimura, H., and Todd. P. (1997) ‘Matching as an Econometric Evaluation Estimator: Evidence from Evaluating a Job Training Program,’ *Review of Economic Studies*, 64, 4, pp. 605-654.

Heckman, J., Ichimura, H., and Todd. P. (1998a) ‘Matching as an Econometric Evaluation Estimator,’ *Review of Economic Studies*, 65, 2, pp. 261-294.

Heckman, J., Ichimura, H., Smith, J., and Todd, P. (1998b) ‘Characterizing Selection Bias Using Experimental Data,’ *Econometrica*, 66, 5, pp. 1017-1098.

Heckman, J., LaLonde, R., and Smith, J. (1999) ‘The Economics and Econometrics of Active Labour Market Programmes,’ in Ashenfelter, O. and Card, D. (eds.), *The Handbook of Labor Economics*, 3, North-Holland, Amsterdam.

Heckman, J. and Singer, B. (1984) ‘Econometric Duration Analysis,’ *Journal of Econometrics*, 24, pp. 63-132.

Heckman, J., Tobias, J., and Vytlacil, E. (2003) ‘Simple Estimators for Treatment Parameters in a Latent Variable Framework,’ *Review of Economics and Statistics*, 85, 3, pp. 748-755.

Heckman, J. and Vytlacil, E. (2000) ‘Instrumental Variables, Selection Models and Tight Bounds on the Average Treatment Effect,’ NBER Technical Working Paper 0259.

Hensher, D. and Bradley, M. (1993) ‘Using Stated Response Data to Enrich Revealed Preference Discrete Choice Models,’ *Marketing Letters*, 4, 2, pp. 139-152.

Hensher, D. and Johnson, N. (1981) *Applied Discrete Choice Modelling*, John Wiley and Sons, New York.

Hensher, D., Rose, J., and Greene, W. (2005) ‘The Implications on Willingness to Pay of Respondents Ignoring Specific Attributes,’ *Transportation*, 32, 3, pp. 203-222.

Hensher, D., Rose, J., and Greene, W. (2015) *Applied Choice Analysis*, 2nd Edition, Cambridge University Press, Cambridge.

Hilbe, J. (2011) *Negative Binomial Regression*, Second Edition, Cambridge University Press, Cambridge.

Hildebrand, G. and Liu, T. (1957) *Manufacturing Production Functions*, Cornell University Press, Ithaca.

Hildreth, C. and Houck, C. (1968) ‘Some Estimators for a Linear Model with Random Coefficients,’ *Journal of the American Statistical Association*, 63, pp. 584-595.

Hodge, A. and Shankar, S. (2014) ‘Partial Effects in Ordered Response Models with Factor Variables,’ *Econometric Reviews*, 33, 8, pp 854-868.

Honore, B. and Kyriazidou, E. (2000) ‘Panel Data Discrete Choice Models with Lagged Dependent Variable Models,’ *Econometrica*, 68, pp. 839-874.

Horn, D., Horn, A., and Duncan, G. (1975) ‘Estimating Heteroskedastic Variances in Linear Models,’ *Journal of the American Statistical Association*, 70, pp. 380-385.

Horowitz, J. (1993) ‘Semiparametric Estimation of a Work-Trip Mode Choice Model,’ *Journal of Econometrics*, 58, pp. 49-70.

Horrace, W. and Schmidt, P. (1996) ‘Confidence Statements for Efficiency Estimates from Stochastic Frontier Models,’ *Journal of Productivity Analysis*, 7, pp. 257-282.

Horrace, W. and Schmidt, P. (2000) ‘Multiple Comparisons with the Best, with Economic Applications,’ *Journal of Applied Econometrics* 15, 1, pp 1-26.

Howe, C., Lee, B., and Bennett, L. (1994) ‘Design and Analysis of Contingent Valuation Surveys Using the Nested Tobit Model,’ *Review of Economics and Statistics*, 76, pp. 385-388.

Hsiao, C. (1986) *Analysis of Panel Data*, Cambridge University Press, Cambridge.

Huang, C. and Liu, T. (1994) ‘Estimation of a Non-Neutral Stochastic Frontier Production Function,’ *Journal of Productivity Analysis*, 5, pp. 171-180.

Hui, W. (1991) ‘Proportional Hazard Weibull Mixtures,’ Mimeo, Department of Economics, Australian National University, Canberra.

Hyslop, D. (1999) ‘State Dependence, Serial Correlation and Heterogeneity in Intertemporal Labor Force Participation of Married Women,’ *Econometrica*, 67, pp. 1255-1294.

Johnson, N. and Kotz, S. (1970) *Distributions in Statistics, Continuous Univariate Distributions - 2*, John Wiley and Sons, New York.

Johnson, N. and Kotz, S. (1974) *Distributions in Statistics - Continuous Multivariate Distributions*, John Wiley and Sons, New York.

Johnson, N. and Kotz, S. (1993) *Distributions in Statistics - Discrete Distributions*, 2nd Edition, John Wiley and Sons, New York.

Johnston, J. and DiNardo, J. (1997) *Econometric Methods*, 4th Edition, McGraw-Hill, New York.

Jondrow, J., Lovell, K., Materov, I., and Schmidt, P. (1982) ‘On the Estimation of Technical Inefficiency in the Stochastic Frontier Production Function Model,’ *Journal of Econometrics*, 19, 2/3, pp. 233-238.

Jones, A., Lomas, J., and Rice, N. (2014) ‘Applying Beta-Type Size Distributions to Healthcare Cost Regressions,’ *Journal of Applied Econometrics*, 29, 4, pp. 649-670.

Joreskog, K. (1973) ‘A General Method for Estimating a Linear Structural Equation System,’ in Goldberger, A. and Duncan, O. (eds.), *Structural Equation Methods in the Social Sciences*, Seminar Press, New York, pp. 85-112.

Judge, G., Hill, C., Griffiths, W., and Lee, T. (1985a) *The Theory and Practice of Econometrics*, John Wiley and Sons, New York.

Judge, G., Hill, C., Griffiths, W., and Lee, T. (1985b) *Introduction to the Theory and Practice of Econometrics*, John Wiley and Sons, New York.

Kalbfleisch, J. and Prentice, R. (1980) *The Statistical Analysis of Failure Time Data,* John Wiley and Sons, New York.

Kennan, J. (1985) 'The Duration of Contract Strikes in U.S. Manufacturing,' *Journal of Econometrics*, 28, pp. 5-29.

Kennedy, W. and Gentle, J. (1980) *Statistical Computing*, Marcel Dekker, New York.

Kiefer, N. (1988) 'Economic Duration Data and Hazard Functions,' *Journal of Economic Literature*, 26, pp. 646-679.

Kim, H. and Pollard, J. (1990) 'Cube Root Asymptotics,' *Annals of Statistics*, pp. 191-219.

Kim, Y and Schmidt, P. (2000) 'A Review and Empirical Comparison of Bayesian and Classical Approaches to Inference on Efficiency Levels in Stochastic Frontier Models with Panel Data,' *Journal of Productivity Analysis*, 14, 2, pp. 91-118.

Klein, R. and Spady, R. (1993) 'An Efficient Semiparametric Estimator for Discrete Choice Models,' *Econometrica*, 61, pp. 387-421.

Kmenta, J. (1967) 'On Estimation of the CES Production Function,' *International Economic Review*, 8, pp. 180-189.

Kodde, D. and Palm, F. (1986) 'Wald Criteria for Jointly Testing Equality and Inequality Restrictions,' *Econometrica*, 54, 5, pp. 1243-1248.

Koenker, R. and Bassett, G. (1978) 'Regression Quantiles,' *Econometrica*, 46, pp. 107-112.

Koenker, R. and Bassett, G. (1982) 'Robust Tests for Heteroscedasticity Based on Regression Quantiles,' *Econometrica*, 50, pp. 43-61.

Koenker, R. and D'Orey, V. (1987) 'Algorithm AS 229: Computing Regression Quantiles,' *Journal of the Royal Statistical Society: Series C* (*Applied Statistics*), 36, 3, pp. 383-393.

Krailo, M. and Pike, M. (1984) 'Conditional Multivariate Logistic Analysis of Stratified Case-Control Studies,' *Applied Statistics*, 44, 1, pp. 95-103.

Krinsky, I. and Robb, L. (1986) 'On Approximating the Statistical Properties of Elasticities,' *Review of Economics and Statistics*, 68, 4, pp. 715-719.

Krinsky, I. and Robb, L. (1990) 'On Approximating the Statistical Properties of Elasticities: Correction,' *Review of Economics and Statistics*, 72, 1, pp. 189-190.

Kumbhakar, S. (1994) Efficiency Estimation in a Profit Maximizing Model Using Flexible Production Function,' *Agricultural Economics*, 10, pp. 143-152.

Kumbhakar, S. (1995) 'Modeling Technical and Allocative Inefficiency in Translog Production Functions,' *Economics Letters*, 63, pp. 12-19.

Kumbhakar, S., Lien, G., and Hardaker, J. (2014) 'Technical Efficiency in Competing Panel Data Models: a Study of Norwegian Grain Farming,' *Journal of Productivity Analysis*, 41, 2, pp. 321–337.

Kumbhakar, S. and Lovell, C. (2000) *Stochastic Frontier Analysis*, Cambridge University Press, Cambridge.

Kumbhakar, S., Parmeter, C., and Tsionas, E. (2013) 'A Zero Inefficiency Stochastic Frontier Estimator,' *Journal of Econometrics*, 172, 1, pp. 66-76.

Kyriazidou, E. (1997) 'Estimation of a Panel Data Sample Selection Model,' *Econometrica*, 65, 1997, pp. 1335-1364.

Lai, H. (2015) 'Maximum Likelihood Estimation of the Stochastic Frontier Model with Endogenous Switching or Sample Selection,' *Journal of Productivity Analysis*, 44, 2, pp. 105-117.

LaLonde, R. (1986) 'Evaluating the Econometric Evaluations of Training Programs with Data,' *American Economic Review*, 76,(4), 1986, pp. 604-620

Lambert, D. (1992) 'Zero-Inflated Poisson Regression, with an Application to Defects in Manufacturing,' *Technometrics,* 34, 1, pp. 1-14.

Lancaster, T. (1985) 'Generalized Residuals and Heterogeneous Duration Models: With Applications to the Weibull Model,' *Journal of Econometrics*, 28, pp. 155-169.

Lancaster, T. (1990) *The Econometric Analysis of Transition Data*, Cambridge University Press, Cambridge.

Land, K., McCall, P., and Nagin, D. (1994) 'Poisson and Mixed Poisson Regression Models: A Review of Applications, Including Recent Developments in Semiparametric Maximum Likelihood Methods,' Manuscript, Department of Sociology, Duke University, Durham.

Land, K., McCall, P., and Nagin, D. (1995) 'A Comparison of Poisson, Negative Binomial and Semiparametric Mixed Poisson Regression Models with Empirical Applications to Criminal Careers Data,' Manuscript, Department of Sociology, Duke University, Durham.

L'Ecuyer, P. (1998) 'Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators,' Working Paper, Department of Information Science, University of Montreal, Montreal.

Lee, B. (1992) 'A Nested Tobit Analysis for a Sequentially Censored Regression Model,' *Economics Letters*, 38, pp. 269-273.

Lee, L. (1976) 'Estimation of Limited Dependent Variable Models by Two-Stage Methods,' Ph.D. Dissertation, University of Rochester, Rochester.

Lee, L. (1978) 'Unionism and Wage Rates: A Simultaneous Equation Model with Qualitative and Limited Dependent Variables,' *International Economic Review*, 19, pp. 415-433.

Lee, L. (1983) 'Generalized Econometric Models with Selectivity,' *Econometrica*, 51, pp. 507-512.

Lee, L., Maddala, G. S., and Trost, R. P. (1980) 'Asymptotic Covariance Matrices of Two Stage Probit and Two Stage Tobit Methods for Simultaneous Models With Selectivity,' *Econometrica*, 48, pp. 491-504.

Lerman, S. and Manski, C. (1981) 'On the Use of Simulated Frequencies to Approximate Choice Probabilities,' in Manski, C. and McFadden, D. (eds.), *Structural Analysis of Discrete Data with Econometric Applications*, MIT Press, Cambridge.

Liang, K. and Zeger, S. (1986) 'Longitudinal Data Analysis Using Generalized Linear Models,' *Biometrika*, 73, pp. 13-22.

Long, S. (1997) *Regression Models for Categorical and Limited Dependent Variables*, Sage Publications, Thousand Oaks.

Longley, J. W. (1967) 'An Appraisal of Least Squares Programs from the Point of the User,' *Journal of the American Statistical Association*, 62, pp. 819-841.

Machado, J. and Santos Silva, J.M.C. (2005) 'Quantiles for Counts,' *Journal of the American Statistical Association*, 100, pp. 1226-1237.

Mroz, T. (1987) 'The Sensitivity of an Empirical Model of Married Women's Hours of Work to Economic and Statistical Assumptions,' *Econometrica*, 55, 4, pp. 765-99.

MacKinnon, J. and White, H. (1985) 'Some Heteroskedasticity Consistent Covariance Matrix Estimators with Improved Finite Sample Properties,' *Journal of Econometrics*, 19, pp. 305-325.

Maddala, G. S. (1983) *Limited Dependent and Qualitative Variables in Econometrics*, Cambridge University Press, Cambridge.

Manski, C. (1975) 'Maximum Score Estimation of the Stochastic Utility Model,' *Journal of Econometrics*, 3, pp. 205-228.

Manski, C. (1985) 'Semiparametric Analysis of Discrete Response: Asymptotic Properties of the Maximum Score Estimator,' *Journal of Econometrics*, 27, pp. 313-333.

Manski, C. (1986) 'Semiparametric Analysis of Binary Response from Response-Based Samples,' *Journal of Econometrics*, 31, pp. 31-40.

Manski, C. (1987) ‘Semiparametric Analysis of Random Effects Linear Models from Binary Panel Data,’ *Econometrica*, 55, pp. 357-362.

Manski, C. (1988) *Analog Estimation Methods in Econometrics*, Chapman and Hall, New York.

Manski, C. and McFadden, D. (eds.) (1981) *Structural Analysis of Discrete Data with Econometric Applications*, MIT Press, Cambridge.

Manski, C. and Thompson, S. (1985) ‘Operational Characteristics of Maximum Score Estimation,’ *Journal of Econometrics*, 32, pp. 85-108.

Manski, C. and Thompson, S. (1987) ‘MSCORE: A Program for Maximum Score Estimation of Linear Quantile Regressions from Binary Response Data with NPREG: A Program for Kernel Estimation of Univariate Nonparametric Regression Functions,’ Department of Economics, University of Wisconsin, Madison.

Matsumoto, M. and Nishimura, T. (1998) ‘Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator,’ *ACM Transactions on Modeling and Computer Simulation*, 8, 1, pp 3-30.

McCullagh, P. and Nelder, J. (1983) *Generalized Linear Models*, Chapman and Hall, New York, p. 137.

McCullough, B. (1996) ‘Consistent Forecast Intervals when the Forecast-Period Exogenous Variables are Stochastic,’ *Journal of Forecasting*, 15, pp. 293-304.

McCullough, B. (1999) ‘Econometric Software Reliability: EViews, LIMDEP, SHAZAM, and TSP,’ *Journal of Applied Econometrics*, 14, 2, pp. 191-202.

McCullough, B. and Renfro, C. (1998) ‘Benchmarks and Software Standards: A Case Study of GARCH Procedures,’ *Journal of Economic and Social Measurement*, 25, pp. 59-71.

McCullough, B. and Vinod, H. (1999) ‘The Numerical Reliability of Econometric Software,’ *Journal of Economic Literature*, 37, 2, pp. 633-665.

McDonald, J. and Moffitt, R. (1980) ‘The Uses of Tobit Analysis,’ *Review of Economics and Statistics*, 62, pp. 318-321.

McFadden, D. (1982) ‘Econometric Models of Probabilistic Choice,’ in Manski, C. and McFadden, D. (eds.), *Structural Analysis of Discrete Data with Econometric Applications*, MIT Press, Cambridge.

McLachlan, C. and Peel, D. (2000) *Finite Mixture Models*, John Wiley and Sons, New York.

Meeusen, W. and van den Broeck, J. (1977) ‘Efficiency Estimation from Cobb Douglas Production Functions with Composed Error, *International Economic Review*, 18, pp. 435-444.

Meng, C. and Schmidt, P. (1985) ‘On the Cost of Partial Observability in the Bivariate Probit Model,’ *International Economic Review*, 26, 1, pp. 71-86.

Mullahy, J. (1986) ‘Specification and Testing of Some Modified Count Data Models,’ *Journal of Econometrics*, 33, pp. 341-365.

Munnell, A. (1990) ‘Why has Productivity Declined? Productivity and Public Investment,’ *New England Economic Review*, pp. 3-22.

Murphy, K. and Topel, R. (2002) ‘Estimation and Inference in Two-Step Econometric Models,’ *Journal of Business and Economic Statistics*, 20, pp. 88-97.

Nagin, D. and Land, K. (1993) ‘Age, Criminal Careers, and Population Heterogeneity: Specification and Estimation of a Nonparametric, Mixed Poisson Model,’ *Criminology*, 31, 3, pp. 327-362.

Nakamura, A. and Nakamura, M. (1983) ‘Part Time and Full Time Work Behavior of Married Women, a Model with a Doubly Truncated Dependent Variable,’ *Canadian Journal of Economics*, 16, pp. 201-218.

Nakosteen, R. and Zimmer, M. (1980) ‘Migration and Income: The Question of Self Selection,’ *Southern Economic Journal*, 46, pp. 840-851.

Nelder, J. and Wedderburn, W. (1972) ‘Generalized Linear Models,’ *Journal of the Royal Statistical Society, A*, 135, pp. 370-384.

Nelson, F. and Olsen, R. (1978) ‘Specification and Estimation of a Simultaneous Equation Model with Limited Dependent Variables,’ *International Economic Review*, 19, pp. 695-710.

Nerlove, M. and Press, J. (1973) ‘Univariate and Multivariate Log-Linear and Logistic Models,’ RAND Corporation Report R-1306-EDA/NIH.

Neumark, D. (1988) ‘Employer’s Discriminatory Behavior and the Estimation of Wage Discrimination,’ *Journal of Human Resources* 23, 3, pp. 279-295.

Newey, W (1984) ‘A Method of Moments Interpretation of Sequential Estimators,’ *Economics Letters*, 14, pp. 201-206.

Newey, W. (1987) ‘Efficient Estimation of Limited Dependent Variable Models with Endogenous Explanatory Variables,’ *Journal of Econometrics*, 36, pp. 231-250.

Newey, W. and West, K. (1987) ‘A Simple Positive Semi-Definite Heteroskedasticity and Autocorrelation Consistent Covariance Matrix,’ *Econometrica*, 55, pp. 703-708.

Oaxaca, R. and Ransom, M. (1994) ‘On Discrimination and the Decomposition of Wage Differentials,’ *Journal of Econometrics* 6, 1, pp. 5-21.

Oaxaca, R. and Ransom, M. (1998) ‘Calculation of Approximate Variances for Wage Decomposition Differential,’ *Journal of Economics and Social Measurement* 24, 1, pp. 55-61.

Olsen, R. (1978) ‘Note on the Uniqueness of the Maximum Likelihood Estimator of the Tobit Model,’ *Econometrica*, 46, pp. 1211-1215.

Olsson, U. (1979) ‘Maximum Likelihood Estimation of the Polychoric Correlation Coefficient,’ *Psychometrika*, 44, 4, pp. 443-460.

Orme, C. (1989) ‘Maximum Likelihood Estimation in the Generalized Logistic Model,’ Mimeo, Department of Economics, University of Nottingham, Nottingham.

Pagan, A. and Vella, F. (1989) ‘Diagnostic Tests for Models Based on Individual Data: A Survey,’ *Journal of Applied Econometrics*, 4, pp. S29-S59.

Papke, L. and Wooldridge, J. (2008) ‘Panel Data Methods for Fractional Response Variables with an Application to Test Pass Rates,’ *Journal of Econometrics*, 145, 1-2, pp. 121-133.

Pence, K. (2006) ‘The Role of Wealth Transformations: An Application to Estimating the Effect of Tax Incentives on Saving,’ *Contributions to Economic Analysis and Policy*, 5, 1, Article 20.

Pesaran, M. (1987) *The Limits to Rational Expectations*, Basil Blackwell, London.

Pesaran, M. and Hall, A. (1988) ‘Tests of Non-Nested Linear Regression Models Subject to Linear Restrictions,’ *Economics Letters*, 27, pp. 341-348.

Petersen, T. (1986a) ‘Fitting Parametric Survival Models with Time-Dependent Covariates,’ *Journal of the Royal Statistical Society*, Series C (Applied Statistics), 35, 3, pp. 281-288.

Petersen, T. (1986b) ‘Estimating Fully Parametric Hazard Rate Models with Time-Dependent Covariates,’ *Sociological Methods and Research*, 14, pp. 219-246.

Phillips, G. and Hale, C. (1977) ‘The Bias of Instrumental Variable Estimators of Simultaneous Equation Systems,’ *International Economic Review*, 18, 1, pp. 219-228.

Phillips, P. and Perron, P. (1988) ‘Testing for a Unit Root in Time Series Regression,’ *Biometrika*, 75, pp. 335-346.

Pindyck, R. and Rubinfeld, D. (1991) *Econometric Models and Economic Forecasts*, 3rd Edition, McGraw-Hill, New York.

Pitt, M. and Lee, L. (1981) ‘The Measurement and Sources of Technical Inefficiency in the Indonesian Weaving Industry,’ *Journal of Development Economics*, 9, pp. 43-64.

Plumper, T. and Troeger, V. (2007) ‘Efficient Estimation of Time-Invariant and Rarely Changing Variables in Finite Sample Panel Analyses with Unit Fixed Effects,’ *Political Analysis*, 15, 2, pp. 124-139.

Plumper, T. and Troeger, V. (2011) ‘Fixed Effects Vector Decomposition: Properties, Reliability and Instruments,’ *Political Analysis*, 19, 2, pp. 147-164.

Poirier, D. (1980) ‘Partial Observability in Bivariate Probit Models,’ *Journal of Econometrics*, 12, pp. 209-217.

Powell, J. (1984) ‘Least Absolute Deviations Estimation for the Censored Regression Model,’ *Journal of Econometrics*, 25, pp. 303-325.

Powell, J. (1986) ‘Symmetrically Trimmed Least Squares Estimation for Tobit Models,’ *Econometrica*, 54, 6, pp. 1435-1460.

Pratt, J. (1981) ‘Concavity of the Log Likelihood,’ *Journal of the American Statistical Association*, 76, pp. 137-159.

Pregibon, D. (1981) ‘Logistic Regression Diagnostics,’ *Annals of Statistics*, 9, pp. 705-724.

Pudney, S. and Shields, M. (2000) ‘Gender, Race, Pay and Promotion in the British Nursing Profession: Estimation of a Generalized Ordered Probit Model,’ *Journal of Applied Econometrics*, 15, 4, pp. 367-399.

Quandt, R. (1983) ‘Computational Problems and Methods,’ in Griliches, Z. and Intrilligator, M. (eds.), *Handbook of Econometrics*, North-Holland, Amsterdam.

Ramsey, J. (1969) ‘Tests for Specification Errors in Classical Linear Least Squares Regression Analysis,’ *Journal of the Royal Statistical Society*, Series B, 31, 2, pp. 350-371.

Reifschneider, D. and Stevenson, R. (1991) ‘Systematic Departures from the Frontier: A Framework for the Analysis of Firm Inefficiency,’ *International Economic Review*, 32, pp. 715-723.

Reimers C. (1983) ‘Labor Market Discrimination Against Hispanic and Black Men,’ *Quarterly Journal of Economics*, 65, pp. 570-579.

Revelt, D. and Train, K. (1998) ‘Mixed Logit with Repeated Choices: Households’ Choices of Appliance Efficiency Level,’ *Review of Economics and Statistics*, 80, pp. 1-11.

Riphahn, R., Wambach, W., and Million, A. (2003) ‘Incentive Effects in the Demand for Health Care: A Bivariate Panel Count Data Estimation,’ *Journal of Applied Econometrics*, 18, 4, pp. 387-405.

Ritter, C. and Simar, L. (1997) Pitfalls of Normal-Gamma Stochastic Frontier Models, *Journal of Productivity Analysis*, 9, pp. 167-182.

Rivers, D. and Vuong, Q. (1988) ‘Limited Information Estimators and Exogeneity Tests for Simultaneous Probit Models,’ *Journal of Econometrics*, 39, pp. 347-366.

Robinson, C. and Tomes, N. (1982) ‘Self Selection and Interprovincial Migration,’ *Canadian Journal of Economics*, 15, pp. 475-502.

Romeu, A. (2004) ‘ExpEnd: Gauss Code for Panel Count-Data Models,’ *Journal of Applied Econometrics*, 19, 3, pp. 429-434.

Rosett, R. and Nelson, R. (1975) ‘Estimation of the Two-Limit Probit Regression Model,’ *Econometrica*, 43, pp. 141-146.

Ruud, P. (2000) *An Introduction to Classical Econometric Theory*, Oxford University Press, Oxford.

Salkever, D. (1976) ‘The Use of Dummy Variables to Compute Prediction Errors and Confidence Intervals,’ *Journal of Econometrics*, 4, pp. 393-397.

Santos Silva, J.M.C. and Windmeijer, F. (2001) ‘Two-Part Multiple Spell Models for Health Care Demand,’ *Journal of Econometrics*, 104, pp. 67-89.

Schmidt, P. (1985) ‘Frontier Production Functions,’ *Econometric Reviews*, 4, 2, pp. 289-328.

Schmidt, P. and Strauss, R. (1975) ‘The Predictions of Occupation Using Multinomial Logit Models,’ *International Economic Review*, 16, 2, pp. 471-486.

Schmidt, P. and Witte, A. (1989) ‘Predicting Criminal Recidivism Using Split Population Survival Time Models,’ *Journal of Econometrics*, 40, pp. 141-159.

Simar, L. and Wilson, P. (1998) ‘Sensitivity Analysis of Efficiency Scores: How to Bootstrap in Nonparametric Frontier Models,’ *Management Science*, 44, pp. 49-61.

Simar, L. and Wilson, P. (1999) ‘Of Course We Can Bootstrap DEA Scores! But, Does It Mean Anything?’ *Journal of Productivity Analysis*, 11, pp. 67-80.

Spector, L. and Mazzeo, M. (1980) ‘Probit Analysis and Economic Education,’ *Journal of Economic Education*, 11, pp. 37-44.

Spitzer, J. (1984) ‘Variance Estimates in Models with the Box-Cox Transformation: Implications for Estimation and Hypothesis Testing,’ *Review of Economics and Statistics*, 66, pp. 645-652.

Staiger, D. and Stock, J. (1997) ‘Instrumental Variables Regression with Weak Instruments,’ *Econometrica*, 65, 3, pp. 557-586.

Stevenson, R. (1980) ‘Likelihood Functions for Generalized Stochastic Frontier Estimation,’ *Journal of Econometrics*, 13, pp. 57-66.

Stewart, M. (1983) ‘On Least Squares Estimation When the Dependent Variable is Grouped,’ *Review of Economic Studies*, 50, pp. 141-149.

Stoker, T. (1986) ‘Consistent Estimation of Scaled Coefficients,’ *Econometrica*, 54, pp. 1461-1482.

Swamy, P. (1971) *Statistical Inference in Random Coefficient Regression Models*, Springer Verlag, New York.

Swamy, P. (1974) ‘Linear Models with Random Coefficients,’ in Zarembka, P. (ed.), *Frontiers in Econometrics*, Academic Press, New York.

Terza, J. (1985) ‘A Tobit-Type Estimator for the Censored Poisson Regression Model,’ *Economics Letters*, 18, pp. 361-365.

Terza, J. (1994) ‘Estimating Count Data Models with Endogenous Switching and Sample Selection,’ Mimeo, Department of Economics, Pennsylvania State University, University Park.

Terza, J. (1998) ‘Estimating Count Data Models with Endogenous Switching: Sample Selection and Endogenous Treatment Effects,’ *Journal of Econometrics*, 84, 1, pp. 129-154.

Terza, J. (2009) ‘Parametric Nonlinear Regression with Endogenous Switching,’ *Econometric Reviews*, 28, pp. 555-580.

Terza, J., Basu, A., and Rathouz , P.J. (2008) ‘Two-stage Residual Inclusion Estimation: Addressing Endogeneity in Health Econometric Modeling,’ *Journal of Health Economics* 27, 3, pp. 531-543.

Train, K. (1998) ‘Recreation Demand Models with Taste Differences over People,’ *Land Economics*, 74, pp. 230-239.

Train, K. (1999) ‘Halton Sequences for Mixed Logit,’ Manuscript, Department of Economics, University of California, Berkeley.

Train, K. (2009) *Discrete Choice Models with Simulation*, 2nd Edition, Cambridge University Press, Cambridge.

Trethaway, M. and Windle, R. (1983) ‘U.S. Airline Cross Section: Sources of Data,’ Mimeo, Department of Economics, University of Wisconsin, Madison.

Tsionas, E. and Kumbhakar, S. (2012) ‘Firm Heterogeneity, Persistent and Transient Technical Inefficiency: A Generalized True Random Effects Model,’ *Journal of Applied Econometrics*, 29, 1, pp. 110–132.

Uebersax, J.S. (2006) ‘The Tetrachoric and Polychoric Correlation Coefficients,’ *Statistical Methods for Rater Agreement web site*. Available at: http://john-uebersax.com/stat/tetra.htm.

Veall, M. and Zimmermann, K. (1992) 'Pseudo-R^2 in the Ordinal Probit Model,' *Journal of Mathematical Sociology*,' 16, pp. 333-342.

Veall, M. and Zimmermann, K. (1996) 'Pseudo-R^2 Measures for Some Common Limited Dependent Variable Models,' *Journal of Economic Surveys*, 10, 3, pp. 241-259.

Verbeek, M. (1990) 'On the Estimation of a Fixed Effects Model with Selectivity Bias,' *Economics Letters*, 34, pp. 267-270.

Verbeek, M. and Nijman, T. (1992) 'Testing for Selectivity Bias in Panel Data Models,' *International Economic Review*, 33, 3, pp. 681-703.

Vuong, Q. (1989) 'Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses,' *Econometrica*, 57, pp. 307-334.

Waldman, D. (1982) 'A Stationary Point for the Stochastic Frontier Likelihood,' *Journal of Econometrics*, 18, pp. 275-279.

White, H. (1978) 'A Heteroskedasticity Consistent Covariance Matrix and a Direct Test for Heteroskedasticity,' *Econometrica*, 46, pp. 817-838.

White, H. (1980) 'Using Least Squares to Approximate Unknown Regression Functions,' *International Economic Review*, 21, 1, pp. 149-170.

White, H. (1981) *Asymptotic Theory for Econometricians*, Academic Press, New York.

Wikstrom, D. (2016) 'Consistent Method of Moments Estimation of the True Fixed Effects Model,' *Economics Letters*, forthcoming.

Williams, R. (2006) 'Generalized Ordered Logit/Partial Proportional Odds Models for Ordinal Dependent Variables,' Department of Sociology, University of Notre Dame, Notre Dame.

Willis, R. and Rosen, S. (1978) 'Education and Self Selection,' *Journal of Political Economy*, 87, pp. S7-S36.

Windmeijer, F. (1995) 'Goodness of Fit Measures in Binary Choice Models,' *Econometric Reviews*, 14, pp. 101-116.

Winkelmann, R. (2008) *Econometric Analysis of Count Data*, 5th Edition, Springer-Verlag, Berlin.

Winkelmann, R. and Zimmermann, K. (1991a) 'A New Approach for Modeling Economic Count Data,' *Economics Letters*, 37, pp. 139-143.

Winkelmann, R. and Zimmermann, K. (1991b) 'Count Data Models for Demographic Data,' Working Paper, SELAPO, University of Munich, Munich.

Winkelmann, R. and Zimmermann, K. (1991c) 'Inference in Misspecified Poisson Models,' Working Paper, SELAPO, University of Munich, Munich.

Wong, W. (1983) 'On the Consistency of Cross-Validation in Kernel Nonparametric Regression,' *The Annals of Statistics*, 11, pp. 1136-1141.

Wong, W. and Famoye, F. (1997) 'Modeling Household Fertility Decisions with Generalized Poisson Regression,' *Journal of Population Economics*, 10, pp. 273-283.

Wooldridge, J. (1995) 'Selection Corrections for Panel Data Models Under Conditional Mean Independence Assumptions,' *Journal of Econometrics*, 68, pp. 115-132.

Wooldridge, J. (1999) 'Asymptotic Properties of Weighted M Estimators for Variable Probability Samples, *Econometrica*, 67, pp. 1385-1406.

Wooldridge, J. (2002) *Econometric Analysis of Cross Section and Panel Data*, MIT Press, Cambridge.

World Health Organization (2000) *The World Health Report, 2000, Health Systems: Improving Performance*, WHO, Geneva.

Wynand, P. and van Praag, B. (1981) 'The Demand for Deductibles in Private Health Insurance,' *Journal of Econometrics*, 17, pp. 229-252.

Yen, S. and Jones, A. (1997) ‘Household Consumption of Cheese: An Inverse Hyperbolic Sine Double-Hurdle Model with Dependent Errors’, *American Journal of Agricultural Economics*, 79, pp. 246-251.

Zabel, J. (1992) ‘Estimating Fixed and Random Effects Models with Selectivity,’ *Economics Letters*, 40, pp. 269-272.

Zamani, H. and Ismail, N. (2011) ‘Functional Form for the Generalized Poisson Regression Model,’ *Communication in Statistics – Theory and Methods*.

Zavoina, R. and McElvey, W. (1975) ‘A Statistical Model for the Analysis of Ordinal Level Dependent Variables,’ *Journal of Mathematical Sociology*, Summer, pp. 103-120.

Zhao, X. and Harris, M. (2004) ‘Demand for Marijuana, Alcohol and Tobacco: Participation, Levels of Consumption, and Cross-Equation Correlations,’ *Economic Record*, 80, 251, pp. 391-410.

LIMDEP 11 *Reference Guide* Index

AIC (Akaike Information Criterion) R-229
Algorithm R-663, R-654,
 BFGS R-674
 Newton's method R-673
APPEND command R-122
ASCII file R-82, R-101, R-107
Autocorrelation R-344, R-437
Average partial effect R-295

Balanced panel R-169
BFGS R-674
BHHH R-436
Binary choice fit R-461
Binary file R-105, R-106
Binary variable R-127
Bivariate normal R-136, R-165, R-455
Bivariate probit R-303
Blanks in file R-109
Block diagonal matrix R-416
Bootstrapping R-533
 correlation R-539
Box-Cox R-371
Box-Cox transformation R-126
Breusch and Pagan test R-344
Butler and Moffitt R-595

CALCULATE command R-445
 correlation coefficients R-466
 random sample R-548
Calculator R-444
Canonical correlation R-383
Categorical variable R-142, R-214, R-287
Censoring R-336
Central limit theorem R-555
Character data R-109
Chi squared R-155
Chow test R-334
CLIST command R-189
CLS R-325
Cluster R-253
 estimator R-254, R-255, R-256
CML R-352
Command R-53
 bar R-80
 builder R-218
 modify R-51
 random parameters R-602
 summary R-69
Commands
 APPEND R-122
 CALCULATE R-445
 CLIST R-189
 CREATE R-124
 DATES R-196
 DECOMPOSE R-321
 DEFAULT R-669
 DELETE R-164
 DISPLAY R-186, R-190, R-233
 DO R-513
 DRAW R-198, R-542, R-677
 FPLOT R-552
 IMPORT R-82, R-84
 IMPUTE R-522
 LIST R-117
 MATRIX R-391
 MERGE R-179
 MPLOT R-401
 NAMELIST R-181
 NOSKIP R-205
 PARTIAL EFFECTS R-263, R-273
 PERIOD R-197
 PROCEDURE R-494
 READ R-101
 RECODE R-161
 REJECT R-193
 RENAME R-164
 SAMPLE R-193
 SETPANEL R-174
 SIMULATE R-312
 SKIP R-204
 SORT R-162
 STRING R-506
 TITLE R-247
 WALD R-360
 WRITE R-121
Confidence interval R-228, R-300
Constant term R-207
Convergence R-502, R-664

Correlation coefficient R-466
  biserial R-466
  polychoric R-469
  rank R-471
  tetrachoric R-467
Covariance matrix R-234, R-358
CREATE
  functions R-133
  matrix R-138
  means R-139
  random sample R-544
CREATE command R-96, R-124, R-128
Critical values R-454
Cross section data R-193
CSV
  file R-83, R-119, R-396
  format R-90
Current sample R-191

Data R-62
  numeric R-87
Data area R-57, R-98
  cells R-99
  rows R-100
  size R-57
Data editor R-93
Data file R-82, R-101, R-110
  blanks R-109
Data matrix R-145, R-422
Data transformation R-124
DATES command R-196
DECOMPOSE command R-321
Decomposition R-320
Decomposition, Oaxaca R-320
DEFAULT command R-669
Default missing value R-92
DELETE command R-164
Delta method R-358
Desktop R-36, R-72
Diagnostics R-698
  unable to find optimum R-658
Dickey Fuller test R-471
Difference in differences R-315
Direct effect R-306
Discrete choice R-337
Discriminant analysis R-385
DISPLAY command R-186, R-190, R-233
DO command R-513
DRAW R-198, R-542, R-677
  panel data R-543
Dummy variable R-134, R-299

Editing window R-40, R-42
Editor R-110
Elasticity R-278
Endogenous binary variable R-474
Endogenous variable R-140, R-476
Equal coefficient vectors R-335
Estimation, suppress results R-240
Excel R-88, R-106, R-112, R-408
EXECUTE R-494, R-523
  conditional R-500
  loop R-508
  query R-499
  quiet R-487
  scalar R-498
  silent R-242
Executing commands R-44, R-485
Exit code R-662
Exit program R-52
Expand R-356
  categorical variable R-142
Exponent R-133
Export
  data R-112, R-119
  results R-248
  variables R-97

File
  CSV R-83, R-396
  data R-82
  Excel R-88, R-106, R-112
  input R-55
  output R-39, R-66, R-246
  path R-43
  project R-38, R-40, R-116
  Stata R-106
  type R-38
Fill R-308
Fitted values R-307
Fixed effects R-562, R-581
  Hausman test R-386
  nonlinear R-590
  saving R-570

  two way R-587
FPLOT command R-552
Gamma distribution R-456
Gamma function R-133, R-456
GEE R-694
Gibbs sampler R-531
GLIM R-694
GMM R-442
GO button R-44, R-48
Graphical results R-240
Group count variable R-178
Group means R-145
Grouped data R-151, R-258
Groupwise heteroscedasticity R-346

Halton sequence R-155, R-617, R-619
Hausman test R-386
Help R-67
Hermite quadrature R-418
Heteroscedasticity R-437
Histogram R-535
Homoscedasticity R-342
Hypothesis test R-234, R-238, R-325

Identity matrix R-416
IMPORT R-82, R-102, R-92
IMPORT command R-82, R-84
Imputation R-518, R-522
  aggregation R-527
  binary variable R-528
  continuous variable R-529
  estimation R-523
  fractional variable R-528
  missing values R-518
  multinomial variable R-529
  ordered outcome R-529
  Poisson variable R-529
IMPUTE command R-522
Incidental parameters problem R-586
INCLUDE R-193, R-677
Indexed variables R-508
Individual specific estimates R-609
Indirect effect R-306
Infinite loop R-515
Information criteria R-229
Initial model results R-231
Input file R-55
Interaction term R-209
  namelist R-213
Iterations R-185, R-654
  convergence R-664
  exit R-657

Jackknife estimation R-539

Kolmogorov Smirnov test R-463
Krinsky and Robb R-358, R-359

Labellists R-189
Labels R-85, R88, R-103
Lagrange multiplier R-380
Lagrange multiplier test R-339
Lags R-137, R-216
Laguerre quadrature R-418
Last model R-250, R-276, R-311
Latent class R-637, R-639
  command R-638
  probability R-642
  posterior probability R-646
Law of large numbers R-555
LCM R-638
Least absolute deviations R-437
Least squares R-231, R-436
Likelihood ratio test R-332, R-337
Linear restrictions R-325, R-328
LIST R-308
LIST command R-117
Listwise deletion R-206, R-530
LM test R-339
  autocorrelation R-344
  groupwise heteroscedasticity R-346
  linear regression R-344
Local variable R-506
Logarithm R-133
Logical expression R-130
Logistic distribution R-134
Lognormal R-155
Looping R-508

Marginal effects R-262
Matrix
  block diagonal R-416
  CALCULATE R-412
  calculator R-389

characteristic root R-426
Cholesky decomposition R-427
correlation R-433
data R-145, R-422
Excel R-408
export results R-396
functions R-138
GMM R-442
Hadamard product R-429
import R-407
inverse R-428
Kronecker product R-429
panel data R-440
partitioned R-415
peek R-396
power R-414
random R-417, R-549
singular value decomposition R-427
square root R-426
statistical output R-399
sum of squares R-434
technical output R-396
work area restore R-402
MATRIX command R-391
Matrix algebra R-379
namelist R-186
Maximum iterations R-657
Maximum likelihood estimator R-229, R-332
constraints R-353
Maximum simulated likelihood R-607
Means R-139
partial effects R-289
Menu R-36
edit R-75
help R-79
insert R-43, R-75
model R-77
project R-76
run R-78
tools R-78
window R-79
Merge, panel data R-176
MERGE command R-179
Mget R-402
Missing data R-191, R-201, R-203, R-206
matrix algebra R-202
SKIP R-204
Missing data code R-91
Missing value code R-92
Missing values R-91, R-96, R-130, R-191, R-204, R-518
CSV files R-91
transformation R-129
Model command R-207
conditional R-232
sample R-223
strata R-223
Model estimates R-232
Model output R-226
confidence interval R-228
Model prediction R-235
Model, information criteria R-229
MPLOT command R-401
Multicollinearity R-355, R-380
Multilevel models R-631
Multiple imputation R-518, R-528
aggregation R-527
estimation R-523
Multivariate normal R-136, R-157, R-165, R-168, R-418, R-455, R-666, R-667
matrix R-159
Murphy and Topel estimator R-473

Namelist R-180
data matrix R-188
deleting R-182
define R-181
display model R-186
editing R-183
indexed R-508
interaction term R-213
matrix algebra R-186
nonlinear terms R-213
wildcard character R-180
NAMELIST command R-181
Names
conventions R-54
matrix R-402
variable R-85, R-87, R-102
Newton’s method R-673
New variable R-95
NIST R-67
Noncentral distribution R-456
Nonlinear function R-209

Nonlinear optimization R-206
Nonlinearities R-185
Nonnumeric R-92
Normal
 bivariate R-165
 truncated R-168
 multivariate R-136, R-157, R-165, R-168, R-418, R-455, R-666, R-667
Normal distribution R-134, R-454
NOSKIP command R-205

Observation labels R-85, R-86, R88, R-103
Observation tags R-86, R-148
 current sample R-196
Observations R-58
Omitted variables R-344
OPG R-436
Optimization R-206, R-653
 diagnostics R-699
 exit code R-662
 premature convergence R-672
 quadrature R-665
 scaling data R-671
 starting values R-670
Order statistics R-460
Output R-226
 matrix R-393
Output file R-39, R-66, R-246
Output window R-39, R-49, R-65, R-117

Panel data R-169, R-562
 balanced R-169, R-564
 DRAW R-200
 fixed effects R-567, R-581
 global setting R-174
 group count variable R-178
 group functions R-145
 group indicator R-171
 group means R-440, R-565
 group size R-173
 invariant variables R-176
 latent class R-567, R-578
 least squares R-581
 matrix R-440, R-442
 maximum likelihood R-585
 merge variables R-176
 missing values R-570
 random coefficients R-567
 random effects R-567
 random parameters R-574
 random sample R-200, R-200
 regression R-562
 unbalanced R-550
Parameters R-377
Partial difference R-216
Partial effects R-237, R-261
 average R-295
 compound model R-302
 confidence interval R-300
 dummy variable R-299
 fixed variables R-286
 issues R-268
 means R-289
 model analyzed R-291
 plotting R-282
 sample partition R-284
 saving R-286
 significance R-300
 single index model R-297
PARTIAL EFFECTS R-311
 sample R-278
 variables R-277
 scenarios R-280
PARTIAL EFFECTS command R-263, R-273
Partial elasticity R-279
Partitioned matrix R-415
PERIOD command R-197
Plot
 matrix R-401
 density R-552
 histogram R-553
Poisson R-156
Pooled model R-251
Posterior probabilities R-646
Predictions R-234, R-307
Probability, multivariate normal R-165
Probit R-444
 homoscedasticity R-342
Procedure R-63, R-484, R-492
 editing R-516
 library R-495
PROCEDURE R-241, R-523
PROCEDURE command R-494
Program parameters, default R-669

Project R-57
Project file R-38, R-40, R-116
Project window R-50, R-59

Quadrature R-418, R-595, R-665
Quiet R-241

Random effects R-549, R-562, R-573, R-593
    Hausman test R-386
    nonlinear R-598
Random matrix R-417
Random number generator R-152, R-153, R-167
    seed R-153, R-544
    uniform R-166
Random parameters R-574, R-600
    distribution R-603
    individual specific estimates R-609
    likelihood function R-616
    simulation R-607
Random sample R-155, R-198, R-542, R-544
    multivariate normal R-546
    panel data R-543
Ranks R-135
READ command R-101
    by variables R-104
Recode character data R-109
RECODE command R-161
REGRESS
    panel data R-582
    fixed effects R-583
REJECT R-91, R-191, R-205
REJECT command R-193
REM R-631
RENAME command R-164
Reserved names R-54
    matrix R-403
    calculator R-449
Residuals R-140, R-235, R-307
Restarting R-64
Restricted least squares R-379
Restrictions R-238, R-325, R-333, R-347
    fixed value R-340, R-348
    nonlinear R-347
Retrievable results R-374
REVIEW R-244
Robust R-251
Robust covariance matrix R-251, R-291
ROWS R-100

SAMPLE R-677
SAMPLE command R-96, R-193
Save R-52
Scenarios, simulate R-313
SCLS estimator R-241
Seed R-153, R-544
SETPANEL R-208
SETPANEL command R-174
SILENT R-242
SIMULATE R-311, R-316
SIMULATE command R-312
Simulation R-312, R-607
    random effects R-549
Single index model R-297
SKIP command R-204
SORT command R-162
Spreadsheet file R-88
Stack data R-145
Standardized variable R-139
Starting values R-608, R-670
Statistic, Wald R-328
STRING command R-506
Stop button R-661
Stratified data R-151, R-258, R-334

Tables of results R-244
Tags R-86, R-148
Test R-328
Text editor R-110, R-484
Time series data R-196
TIMER R-230
TITLE command R-247
Toolbar R-37, R-80
Trace R-65
Transformation R-124
    algebraic R-126
    conditional R-130
    if/else R-130
Transition matrix R-287
Treatment effects R-315
Trend variable R-129, R-134
Truncated normal R-168, R-455
Tutorial R-46
Two part model R-336

Two step estimation R-473

Variable, Lhs and Rhs R-207
Variable names R-85, R-87, R-102
    compound R-159
Variance, sample R-168

WALD R-264, R-361
    recursive function R-363
    results R-362
WALD command R-360
Wald statistic R-328
Warning message R-657
Weights R-207, R-225, R-291
WRITE command R-121
    data file R-121
WTP R-372