LIMDEP

Version 9

Student 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
215 Excelsior Avenue
Castle Hill, NSW 2154
Australia
Tel: +61 (0)4-1843-3057
Fax: +61 (0)2-9899-6674
Email: hgroup@optusnet.com.au

End-User License Agreement

This is a contract between you and Econometric Software, Inc. The *software product* refers to the computer software and documentation as well as any upgrades, modified versions, copies or supplements supplied by Econometric Software. By installing, downloading, accessing or otherwise using the software product, you agree to be bound by the terms and conditions of this agreement.

Copyright, Trademark, and Intellectual Property

This software product is copyrighted by, and all rights are reserved by Econometric Software, Inc. No part of this software product, either the software or the documentation, may be reproduced, distributed, downloaded, stored in a retrieval system, transmitted in any form or by any means, sold or transferred without prior written permission of Econometric Software. You may not modify, adapt, translate, or change the software product. You may not reverse engineer, decompile, dissemble, or otherwise attempt to discover the source code of the software product.

LIMDEP™ and *NLOGIT*™ are trademarks of Econometric Software, Inc. The software product is licensed, not sold. Your possession, installation and use of the software product does not transfer to you any title and intellectual property rights, nor does this license grant you any rights in connection with software product trademarks.

Use of the Software Product

You have only the non-exclusive right to use this software product. A single user license is registered to one specific individual, and is not intended for access by multiple users on one machine, or for installation on a network or in a computer laboratory. For a single user license only, the registered single user may install the software on a primary stand alone computer and one home or portable secondary computer for his or her exclusive use. However, the software may not be used on the primary computer by another person while the secondary computer is in use. For a multi-user site license, the specific terms of the site license agreement apply for scope of use and installation.

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 express 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 general, integrated computer package for estimating the sorts of econometric models that are most frequently analyzed with cross section and panel data. Its range of capabilities include basic linear regression and descriptive statistics, the full set of techniques normally taught in the first year of an econometrics sequence, and a tremendous variety of advanced techniques such as nested logit models, parametric duration models, Poisson regressions with right censoring and nonlinear regressions estimated by instrumental variables and the generalized method of moments (GMM). *LIMDEP*'s menu of options is as wide as that of any other general purpose program available, though, as might be expected, longer in some dimensions and shorter in others. Among the signature features of *LIMDEP* is that you will find here many models and techniques that are not available in any other computer package. In addition to the estimation programs you need for your model building efforts, you will also find in *LIMDEP* all the analytical tools you need, including matrix algebra, a scientific calculator, data transformation features and programming language elements, to extend your estimators and create new ones.

This program has developed over many years (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 9.0 continues our periodic cycle of collecting, then incorporating the many suggestions we receive from our users. We also update *LIMDEP* every few years to incorporate new developments in econometrics and to meet the changing demands of our users. As you will see when you read our description of what's new in this version, a major theme of this revision is panel data. Panel data sets and techniques are becoming ever more popular. We are confident that with this revision of *LIMDEP*, you will now have in hand a collection of techniques and procedures for analyzing panel data that is without parallel in any other computer program available anywhere.

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.

William H. Greene
Econometric Software, Inc.
15 Gloria Place
Plainview, NY 11803
January 2007

Preface to the LIMDEP 9.0 Student User's Guide

This user's guide is constructed specifically for the student who is using *LIMDEP* for the first time and is, most likely. It incorporates the *LIMDEP* Reference Guide and the *LIMDEP* Econometric Modeling Guide. The *LIMDEP Reference Guid*e chapters (1-11) discuss operation of the software. There will be several examples drawn largely from econometrics to illustrate the operation of the program. However, the purpose of these chapters are to show you how to use the software, rather than to show you specifically how to estimate and analyze econometric models with the program. In these chapters, I, we are concerned with features such as how to read a data set, how to transform data, and what you need to know about missing data.

The *LIMDEP Econometric Modeling Guide* chapters (12-16) discuss econometrics. These chapters follow essentially the sequence of topics that one might encounter in an econometrics course. Thus, they will describe your data, then describe how to use the linear regression model. This topic usually takes most of the first semester, and it occupies a large section of this part of the manual. We then proceed to some of the more advanced topics that would logically appear later in the course, such as two stage least squares (instrumental variable estimation) and basic discrete (binary) choice models. The last chapter offers a brief survey of the many somewhat and extremely advanced features that are also available in *LIMDEP*, but not covered in a conventional econometrics course (or in this manual).

Having introduced the manual as above, we do emphasize, this user's guide is not an econometrics or statistics text, and does not strive to be one. The material below will present only the essential background needed to illustrate the use of the program. In order to accommodate as many readers as possible, we have attempted to develop the material so that it is accessible to both undergraduates and graduate students. (For the latter, a text that would be useful to accompany this guide is *Econometric Analysis, 7th Edition* (William Greene, Prentice Hall, 2011), which was written by the author of both *LIMDEP* and this manual.)

Table of Contents

Chapter 1: Introduction to *LIMDEP*

1.1 The *LIMDEP* Program

LIMDEP is an integrated package for estimating and analyzing econometric models. It is primarily oriented toward cross section and panel data. But, many standard problems in time series analysis can be handled as well. *LIMDEP*'s basic procedures for data analysis include:

- descriptive statistics (means, standard deviations, minima, etc.), with stratification,
- multiple linear regression and stepwise regression,
- time series identification, autocorrelations and partial autocorrelations,
- cross tabulations, histograms, and scatter plots of several types.

You can also model many extensions of the linear regression model 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, 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,
- 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,
- 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 25 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.

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 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.

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 15 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, a feature not available in any other package.

1.2 References for Econometric Methods

This manual will document how to use LIMDEP for econometric analysis. There will be a number of examples and applications provided as part of the documentation. However, we will not be able to provide extensive background for the models and methods. A few of the main general textbooks currently in use are:

- Baltagi, B., *Econometric Analysis of Panel Data*, 3rd ed., Wiley, 2005
- Cameron, C. and Trivedi, P., *Microeconometrics: Methods and Applications*, Cambridge University Press, 2005.
- Greene, W., *Econometric Analysis*, 6th Edition, Prentice Hall, 2008.
- Gujarati, D., *Basic Econometrics*, McGraw Hill, 2003.
- Johnston, J. and DiNardo, J., *Econometric Methods*, 4th Edition, McGraw-Hill, 1997.
- Stock, J. and Watson, M., *Introduction to Econometrics*, 2nd. Ed., Addison Wesley, 2007.
- Wooldridge, J., *Econometric Analysis of Cross Section and Panel Data*, MIT Press, 2002.
- Wooldridge, J., *Modern Econometrics*, 2nd ed., Southwestern, 2007

Chapter 2: Getting Started

2.1 Introduction

This chapter will describe how to install *LIMDEP* on your computer. Sections 2.5 to 2.7 describe how to start and exit *LIMDEP*.

2.2 Equipment

LIMDEP is written for use on Windows driven computers. As of this writing, we do not support operation on any Apple Macintosh computers. Windows emulation software that allows you to run Windows on Apple machines should allow *LIMDEP* to operate, but we are unable to offer any assurance, nor any specific advice.

2.3 Installation

To install *LIMDEP*, first close all applications. Insert the *LIMDEP* CD in your CD-ROM drive. The setup program should start automatically. If it does not, open My Computer or Windows Explorer (see Figure 2.2) and double click your CD-ROM drive to view the contents of the *LIMDEP* CD. To launch the Setup program, double click Setup. In either case, the installation wizard will start.

Installation proceeds as follows:

Step 1. Preparing to Install…: The InstallShield Wizard checks the operating system and configures the Windows installer. Installation proceeds to the next step automatically.

Step 2. Welcome to the InstallShield Wizard for *LIMDEP* 9.0: Note, *LIMDEP* is protected by copyright law and international treaties. Click Next to continue.

Step 3. Econometric Software End-User License Agreement: In order to install *LIMDEP* 9.0, you must select 'I accept the terms in the license agreement.' Click Next to continue.

Step 4. Welcome to *LIMDEP* 9.0: This window presents a brief introduction to *LIMDEP* 9.0. Please take a moment to read this information and click Next to continue.

Step 5. Destination Folder: This window indicates the destination folder for the program: C:\Program Files. (The complete folder location is C:\Program Files\Econometric Software\LIMDEP9\Program.) Click Next to continue.

Step 6. Ready to Install the Program: This window reviews the installation instructions. Click Install to install *LIMDEP* 9.0. Installation takes about 60 seconds.

Step 7. InstallShield Wizard Completed: Click Finish to complete installation. (You do not have to restart your computer to complete installation.)

Step 8. The first time you launch *LIMDEP*, you will be presented with the Welcome and Registration dialog box. See Section 2.3.1 for a complete explanation of the registration process.

Installation also creates a resource folder, C:\LIMDEP9 with three subfolders:

C:\LIMDEP9\Data Files
C:\LIMDEP9\LIMDEP Command Files
C:\LIMDEP9\Project Files

NOTE: All sample data files referenced in this documentation will be found in these folders.

2.4 Registration

The first time you use *LIMDEP* you will be presented with the Welcome and Registration dialog box. There are two steps to register *LIMDEP*. First, provide the registration information requested in the dialog box. Carefully input the serial number included with your program. This will place the registration information, including your serial number, in the About box. You must complete all three fields of this dialog box in order to begin using *LIMDEP*. See Figure 2.1.

Second, send your registration information to Econometric Software. You can register with Econometric Software by completing the registration card included with your order and faxing or mailing it to us. You can also send your registration information to Econometric Software online via our website, www.limdep.com. To submit your registration information on our website, click Help, then select LIMDEP Web Site and proceed to the Registration page.

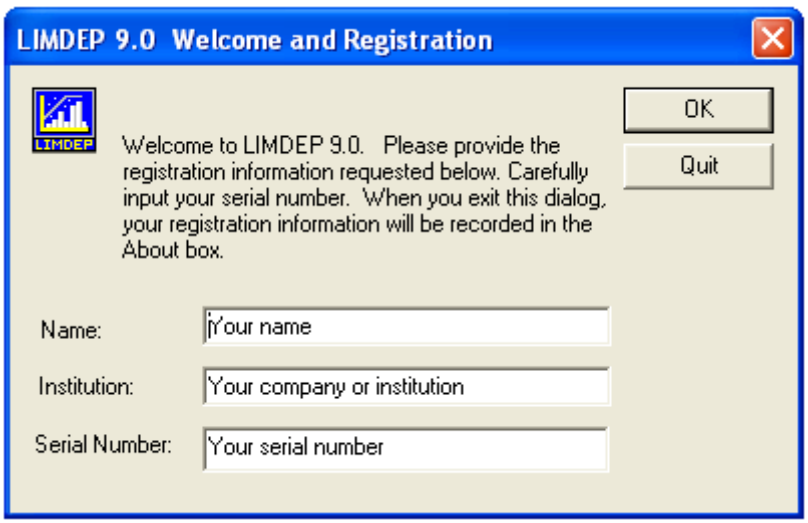


Figure 2.1 Welcome and Registration Dialog Box

2.5 Execution – Beginning the *LIMDEP* Session

Start *LIMDEP* as you would any other program, for example by double clicking the *LIMDEP* icon on your desktop.

The opening window is the *LIMDEP* desktop, shown in Figure 2.2. At the top of the screen, the main menu is shown above the *LIMDEP* toolbar. Below the toolbar is the command bar discussed in Chapter 3. The open window is the project window. The session is identified as your 'project,' which will ultimately consist of your data and the various results that you accumulate. This is where you will begin your *LIMDEP* session. Operation is discussed in Chapter 3.

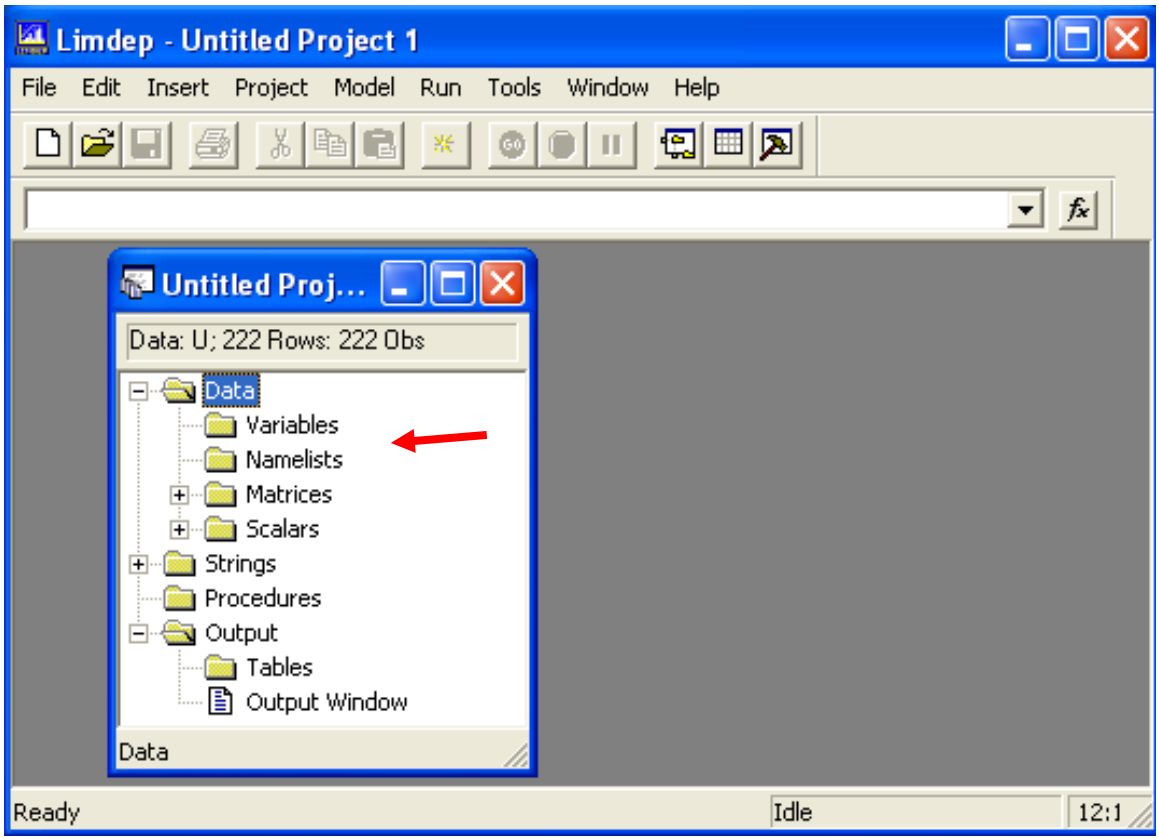


Figure 2.2 *LIMDEP* Desktop Window

If you do not see the command bar when you first start the program, so that your initial desktop appears as below in Figure 2.3, then select Tools in the desktop menu and Options... from the drop down menu. In the dialog that appears next, as shown in Figure 2.4, tick the option to "Display Command Bar" to change the desktop menu so that it will then appear as in Figure 2.2, with the command bar included.

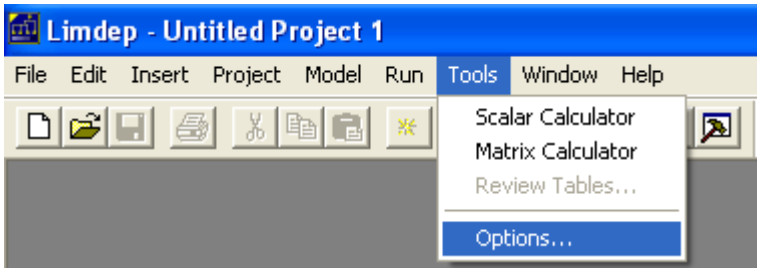


Figure 2.3 Tools Menu on Desktop

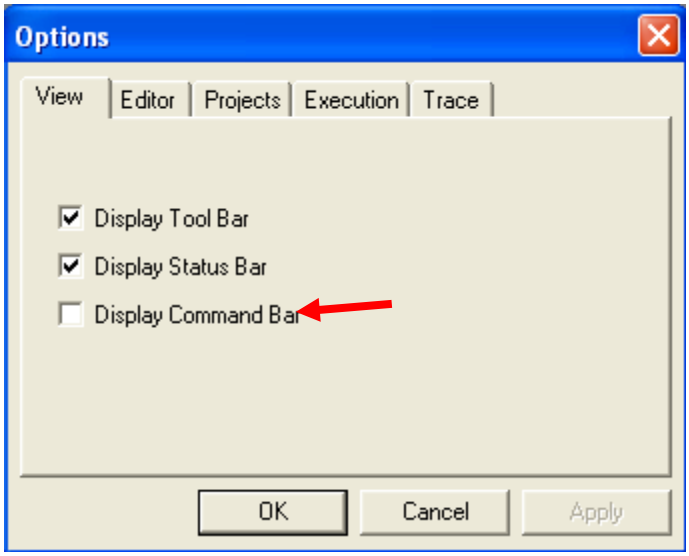


Figure 2.4 Options Dialog Box

2.6 Components of a *LIMDEP* Session

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

- Your data, matrices, scalars, the environment, and so on. The window associated with this information is the project window – usually at the upper left of your screen as in Figure 2.2..
- 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 file described in Section 2.11.

Figure 2.5 shows an example. In this session, we are analyzing a data set that contains 840 observations on about 20 variables. The chapters to follow will describe the various components and how the analysis proceeds. Figure 2.5 shows a fairly typical arrangement of a *LIMDEP* session. (The screen parts are arranged for the figure. It will be more conveniently spaced when you use the program.)

The four parts of the session can be seen in the figure:

- The project consists of the CLOGIT data, which we will use later in several examples.
- The editing window (also referred to as a text or command editor) at the upper right shows the one command that we have entered.
- The output window (mostly obscured) is in the lower half of the split window.
- The trace of the session is shown in the center of the screen, at the top of the split output window.

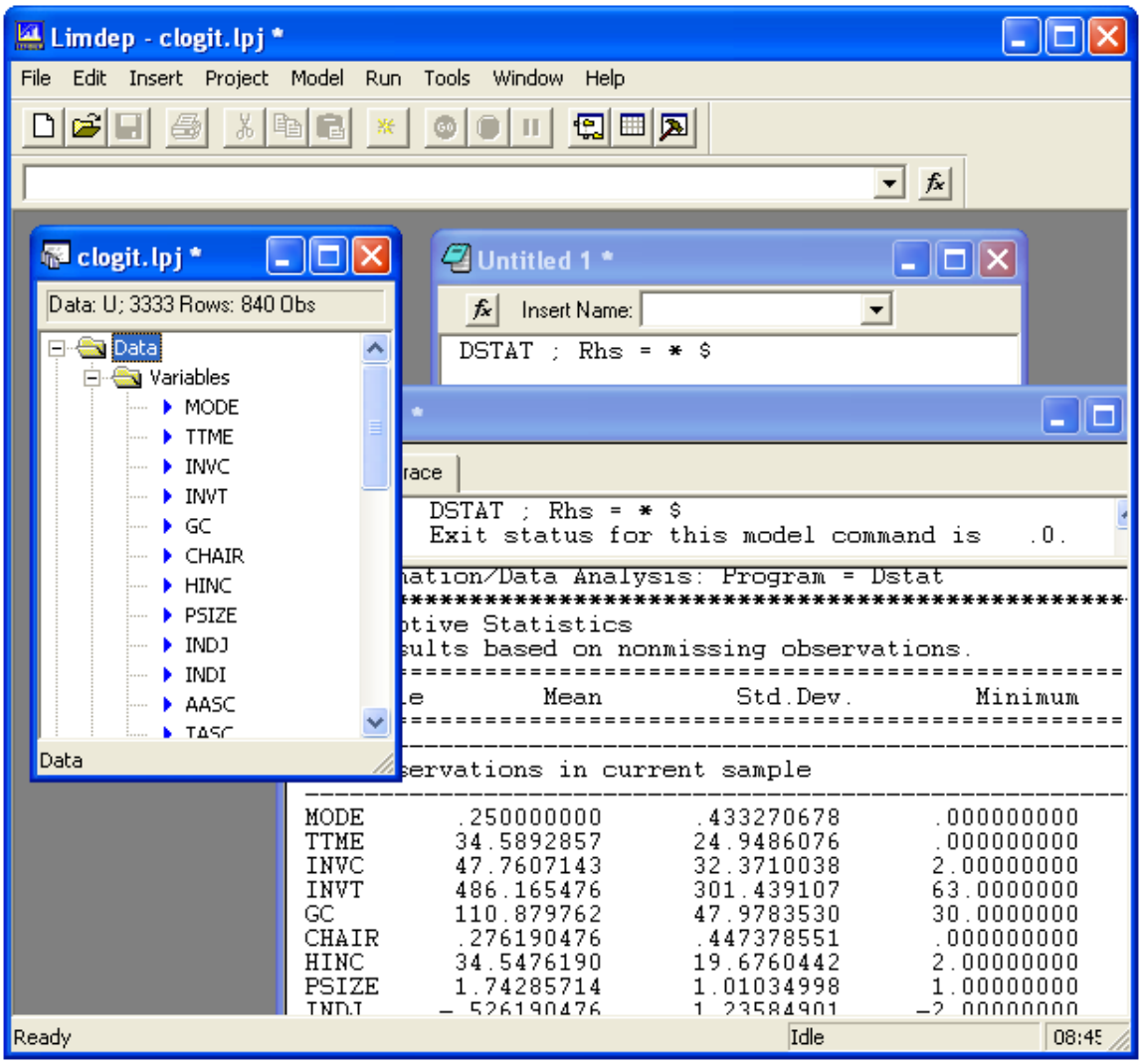


Figure 2.5 *LIMDEP* Data Analysis Session

2.7 Exiting *LIMDEP* and Saving Results

To leave *LIMDEP*, select Exit from the File menu. 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.

When you exit, *LIMDEP* will 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 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 2.6 for an example, where the query refers to the output window. You may also have other working windows open, such as graphs or your scientific calculator and, if so, you can save these as well. This operation is discussed further in Chapter 3.

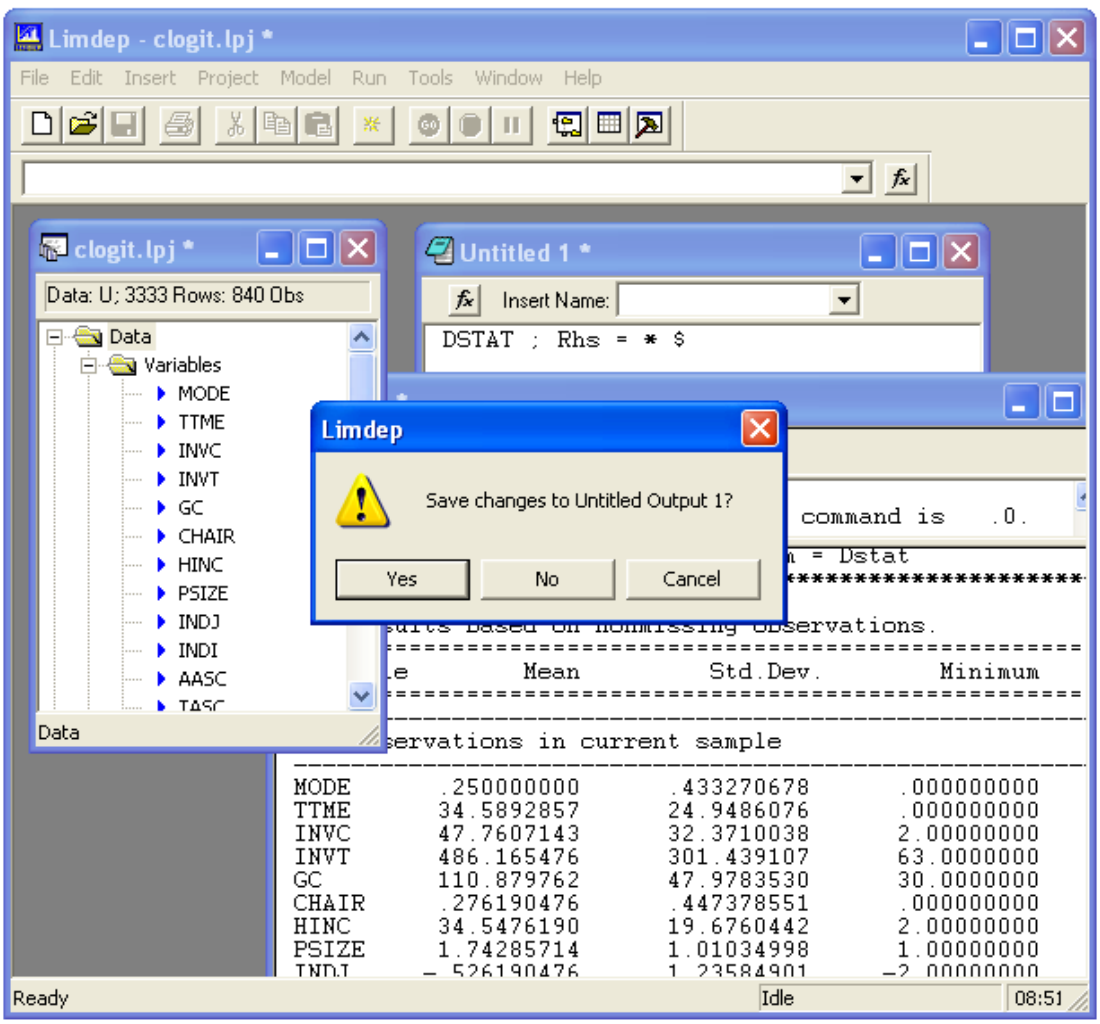


Figure 2.6 Exiting *LIMDEP* – Saving Window Contents

The filename extension for a saved project window is .LPJ. The extension for a saved editing window file or output window is .LIM. 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 1-4 are the last four editing or output window files saved by *LIMDEP*, and the files listed 5-8 are the last four project files. (See Figure 2.7.) Just click the file name in the File menu to open the file. You can also save files by using the save options in the File menu.

NOTE: A saved editing window is referred to as a command or input file.

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.

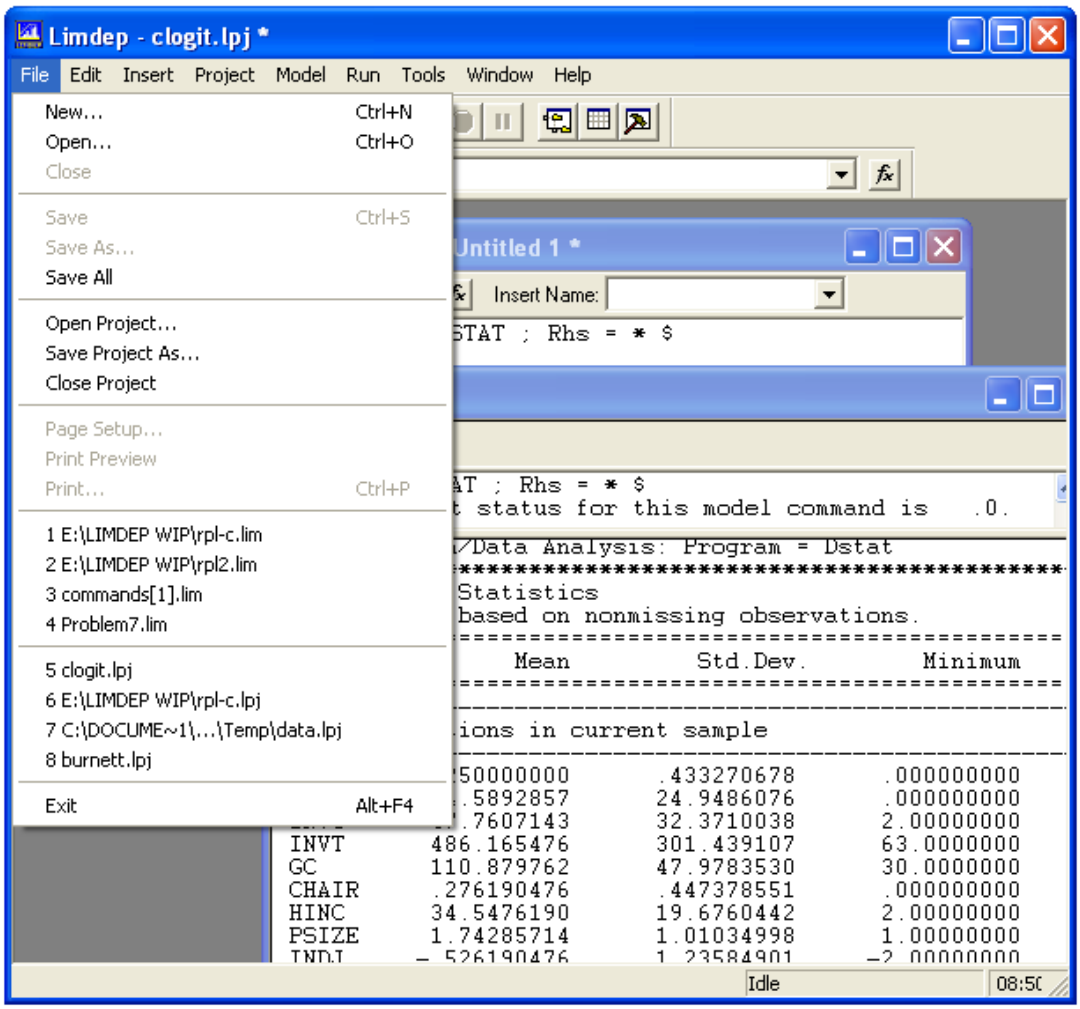


Figure 2.7 File Menu

2.8 Fast Input of a Data Set with OPEN/LOAD

File:Save (or just saving the files upon Exit) then File:Open offers an extremely fast and easy way for you to transform ASCII or any other raw format data to binary format for quick input. The first time you use the data set, use File:Save or the *LIMDEP* **SAVE** command – see Chapter 3 – to rewrite it in *LIMDEP*'s binary format. (You can create any new variables you want to before doing so, these will be written with the raw data.) Later, instead of reading the raw data set, just use File:Open or any of the Windows Explorers to retrieve the file you saved earlier. If you created any new variables before you saved the data set, they will be loaded as well. The time savings in reading an ASCII data set which is unformatted are about 90%! For a formatted data set, it is still at least three times as fast.

TIP: The popular data transfer program *Stat/Transfer* supports *LIMDEP* and will translate many common system files, such as *SAS, Stata, SYSTAT, TSP, EViews*, and so on to .LPJ style *LIMDEP* project files.

2.9 Starting *LIMDEP* from your Desktop or with a Web Browser

Files with the .LPJ and .LIM extensions are 'registered' in your Windows system setup. This means that whenever you double click any file with a .LPJ or .LIM extension in any context, such as Find File, My Computer, any miniexplorer, or in a web browser, Windows will launch *LIMDEP*, and open the file. Note, however, in order to operate *LIMDEP*, you must have a project file open, not just a command or input file.

For example, you can create shortcuts by moving any .LIM or .LPJ files you wish to your Windows desktop. These files will then appear as icons on your desktop, and you can launch *LIMDEP* from your desktop. Similarly, if you open a .LIM file on our website or someone else's, or a .LIM file that is sent to you as an email attachment, you can launch *LIMDEP* and place the indicated file in an editing window. If you start *LIMDEP* in this way, you must then use File:New/Project to open a new project window.

NOTE: Until you open a project, no other program functions are available. You must now open a project with any of the options in the File menu in order to proceed.

Chapter 3: Operating *LIMDEP*

3.1 Introduction

This chapter will explain how to give commands to *LIMDEP* and will describe some essential features of operation. The sections to follow are:

3.2: Beginning the *LIMDEP* Session
3.3: Using the Editing Window
3.4: A Short Tutorial
3.5: Help

3.2 Beginning the *LIMDEP* Session

When you begin your *LIMDEP* session from the Start:Programs menu or a shortcut on your desktop, the initial screen will show a project window entitled 'Untitled Project 1' and an empty desktop as shown in Figure 3.1. You can now begin your session by starting a new project or reloading an existing one. Figure 3.2 shows the File menu (the lower sections show some of our previous work).

3.2.1 Opening a Project

You can select Open or Open Project in the File menu (they are the same at this point) to reload a project that you saved earlier or select one of the existing projects known to *LIMDEP* (if any are). You may also select New to begin a new project.

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.

Note that the window shows that the data area has 3,333 rows. Your project window will show a different value. This is determined by a setting of the size of the data area

TIP: You can associate a *LIMDEP* project file (see Section 2.9) with the program, and launch *LIMDEP* directly with your project file. Use My Computer or the Windows Explorer to navigate to the folder where you have created your project file – its name will be <the name>.LPJ. Drag the icon for the .LPJ file to your desktop, then close My Computer. Now, you can double click the icon on the desktop to launch *LIMDEP* and open the project file at the same time to begin your session.

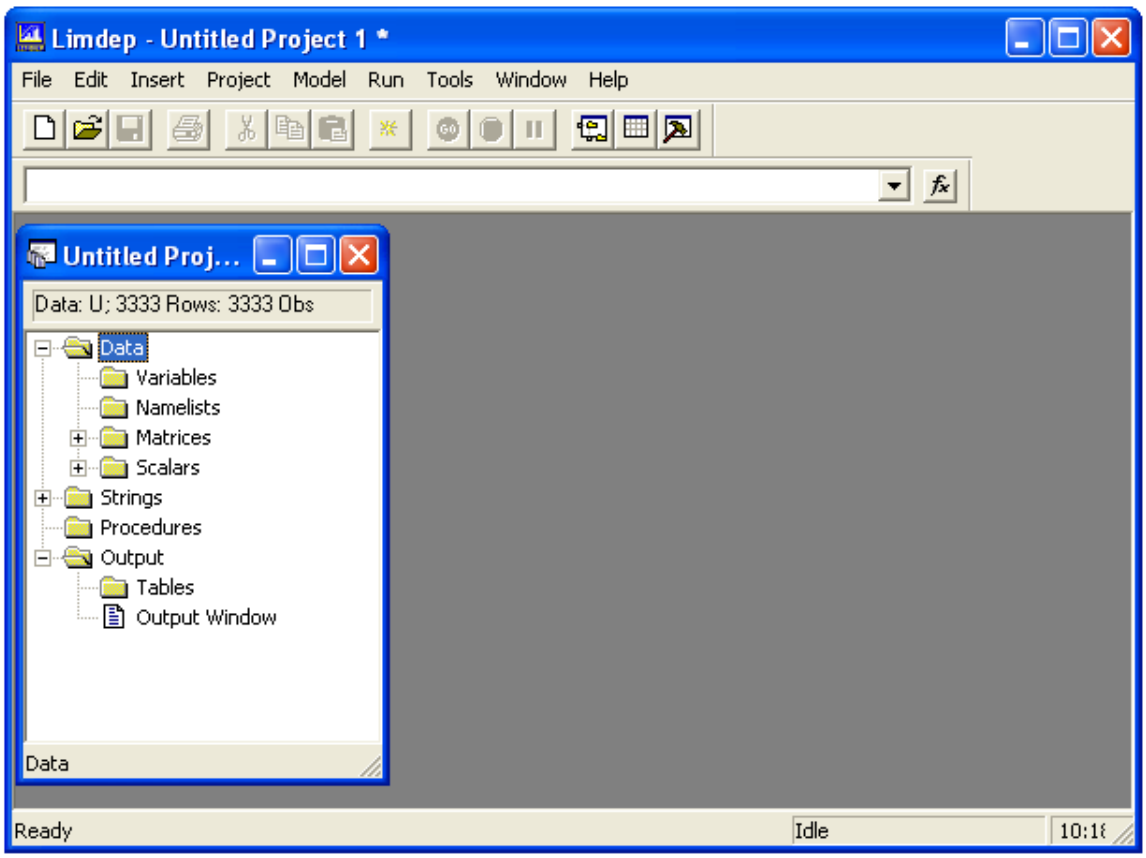


Figure 3.1 The *LIMDEP* Desktop

3.2.2 Opening an Editing Window

To begin entering commands you should now open an editing window. Select New in the File menu to open the File:New dialog box as shown in Figure 3.3. Now, select Text/Command Document and OK to open the editing window, which will appear to the right of the project window.

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

The desktop will now appear as shown in Figure 3.4, and you can begin to enter your commands in the editing window as we have done in an example in the figure. (We have arranged the various windows for appearance in our figures. Your desktop will be more conveniently arranged, and will be full screen sized.)

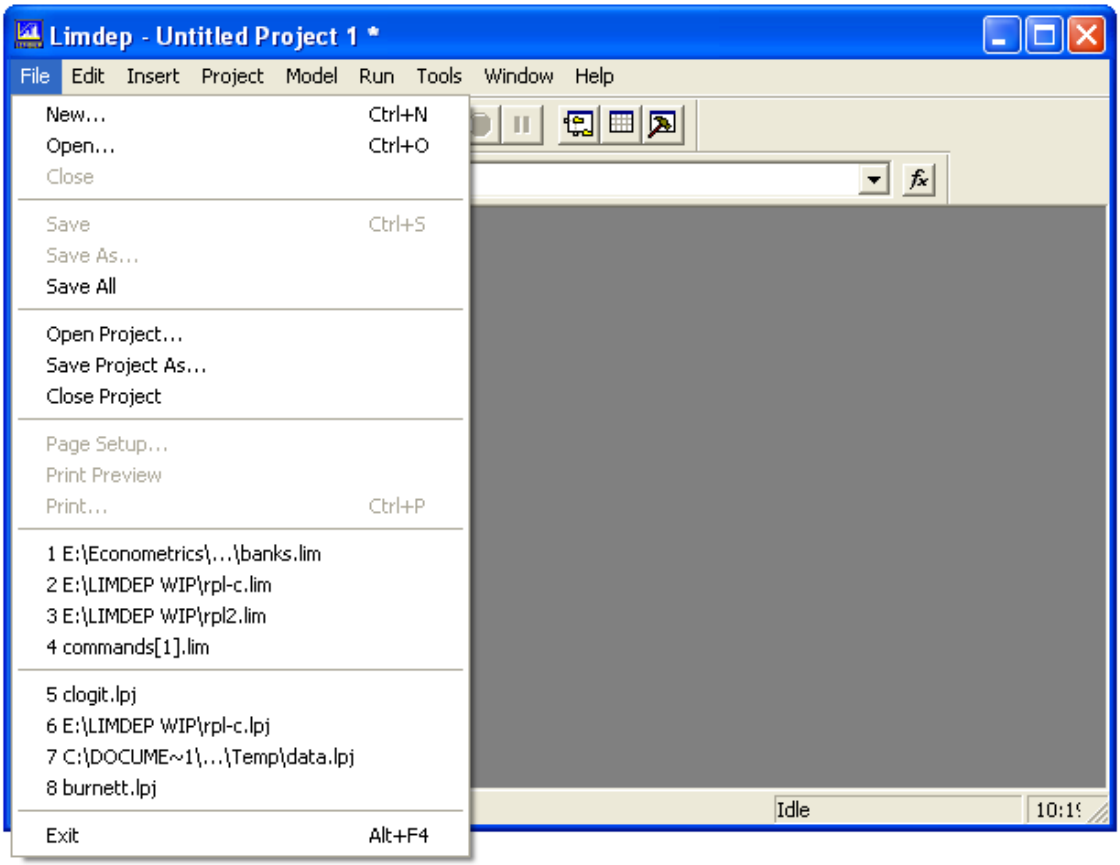


Figure 3.2 The File Menu

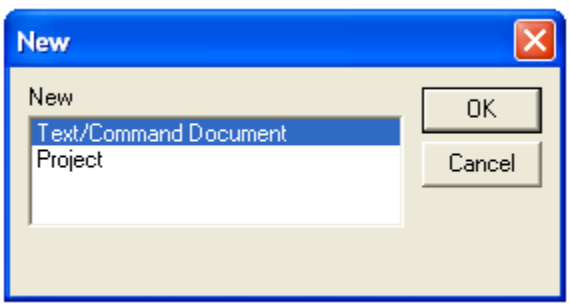


Figure 3.3 The File:New Dialog Box

NOTE: If you have created a text file that contains *LIMDEP* commands that you will be using, instead of creating a new set of commands, you can use File:Open to open that file. If the file that you open has a .LIM file extension in its filename, then *LIMDEP* will automatically open an editing window and place the contents of the file in the window. You can also use Insert:Text File to place a copy of a text file (any text file) in an editing window that is already open.

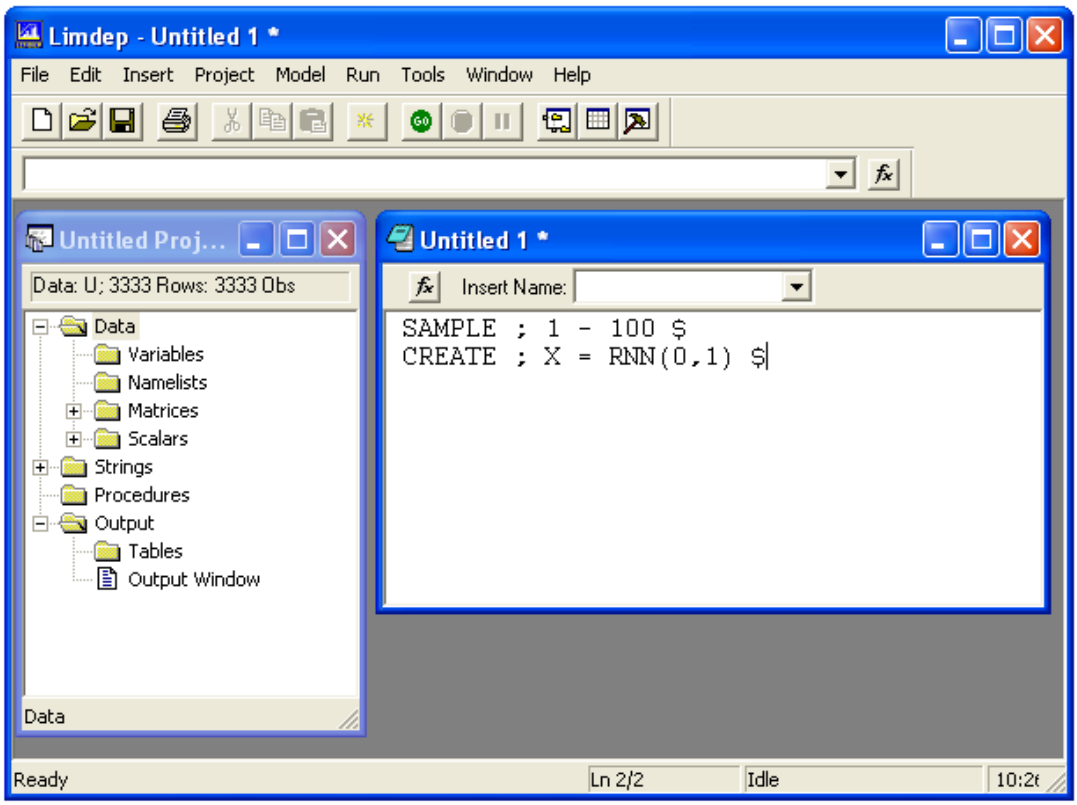


Figure 3.4 Project Window and Editing Window

Note that the editing window shown in Figure 3.4 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.

TIP: You can also associate a *LIMDEP* command file, <the name>.LIM, with *LIMDEP*. As in the earlier tip, if you use My Computer and your mouse to drag the icon for a .LIM file to the desktop, then you can double click the icon to start *LIMDEP* and at the same time, open an editing window for this command file. Note, however, that when you do this, you must then either open an existing project file with File:Open, File:Open Project, or one of the menu entries, or start a new project with File:New/Project/OK.

3.3 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 Undo, Cut, Copy, Paste, Clear, Select All, Find, Replace, and so on, as shown in Figure 3.5. 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 and 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.

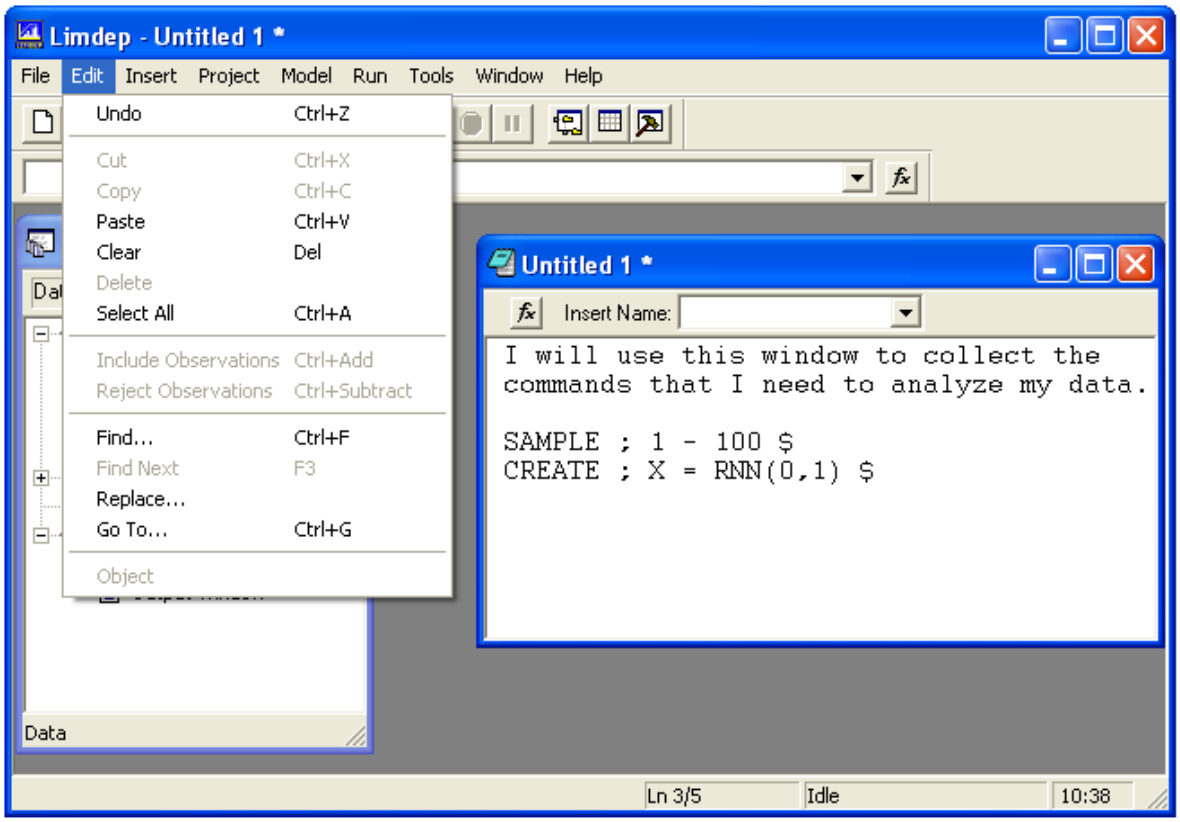


Figure 3.5 Editing Window and the Edit Menu

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.

NOTE: The editing window is also referred to as a text editor or a command editor.

When you are ready to execute commands, highlight the ones you wish to submit with your mouse by placing the cursor at the beginning of the first line you wish to submit and, while holding down the left mouse button, moving the mouse cursor to the end of the last line you wish to submit. (This is the same movement that you use in your word processor to highlight text.) The highlighted section will change from black text on white to white text on black. Then, to execute the commands you may do either of the following:

- Click GO on the *LIMDEP* toolbar. (See  below. If the toolbar is not showing on your screen, select the Tools:Options/View tab, then turn on the Display Tool Bar option.) The GO button is seen in green in the tool bar in Figure 3.6.

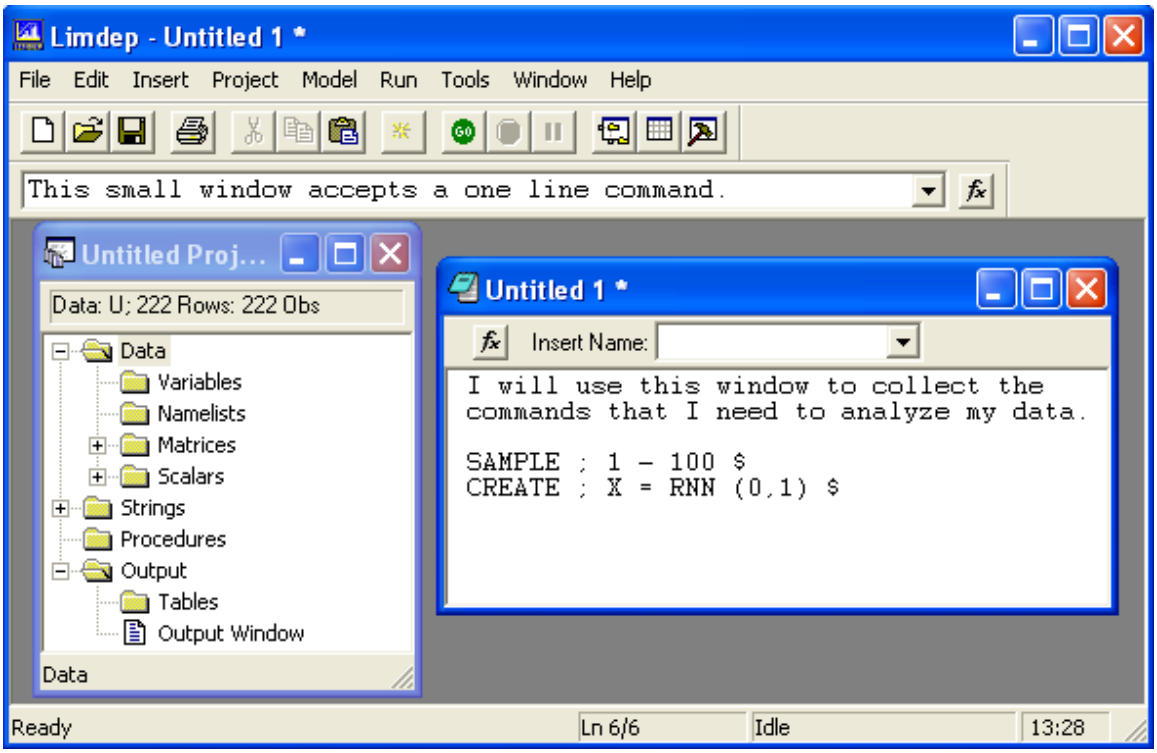


Figure 3.6 Command Bar

- Select the Run menu at the top of your screen. The Run menu is shown in Figure 3.7. The first two items in this menu are:

  - Run Line (or Run Selection if multiple lines are highlighted) will allow you to execute the selected commands once.
  - Run Line Multiple Times (or Run Selection Multiple Times if multiple lines are highlighted) will allow you 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 3.4.

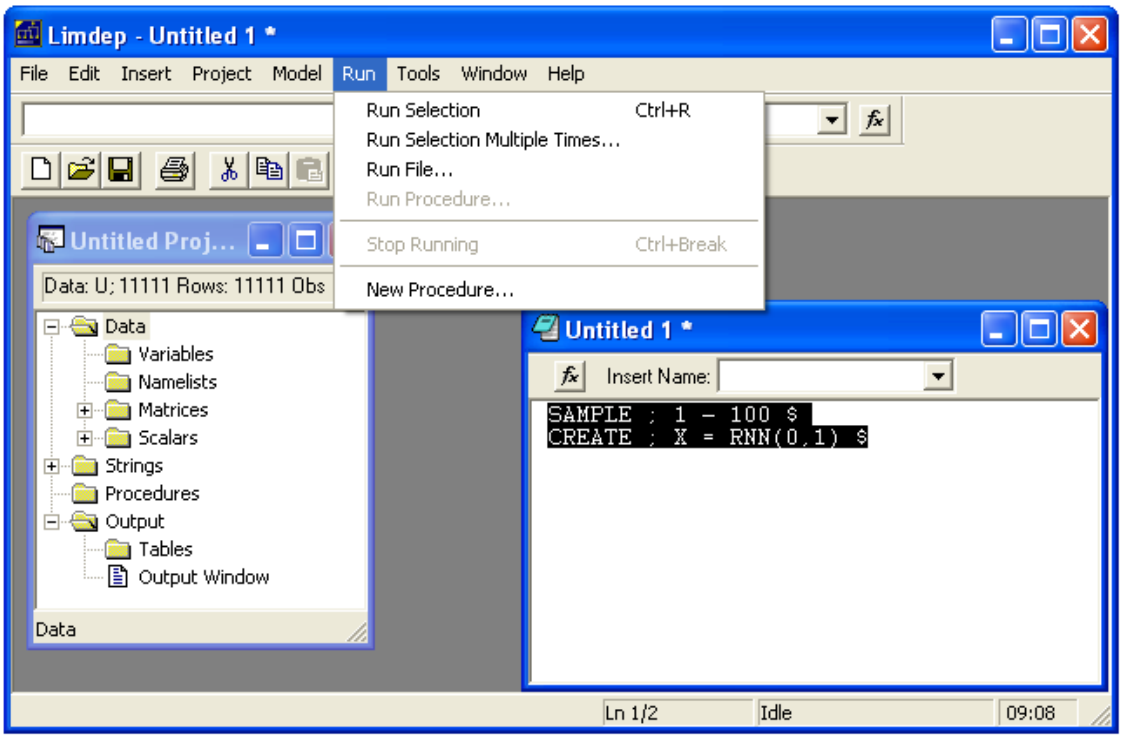


Figure 3.7 Submitting Commands with the Run Menu

If your commands fit on a single line – many of *LIMDEP*'s commands do not, there are additional ways for you to submit commands:

- You can type a one line command in the command window, then press Enter to submit it. The command window or command bar is the small window located below the *LIMDEP* toolbar. The ▼ button at the end of the line allows you to recall and select among the last several such commands you have submitted. See Figure 3.6.

- You can submit a single line of text in the text editor to the command processor 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.

- For desktop computers (*this does not apply to most laptop or notebook computers*), the two Enter keys on your keyboard, one in the alphabetic area and one in the numeric keypad, are different from *LIMDEP*'s viewpoint. You can submit the line with the cursor in it as a one line command by pressing the *numeric* Enter key. The *alphabetic* Enter key acts like an ordinary editing key.

Finally, the right mouse button is also active in the editing window. The right mouse button invokes a small menu that combines parts of the Edit and Insert menus, as shown in Figure 3.8. 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, and 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.

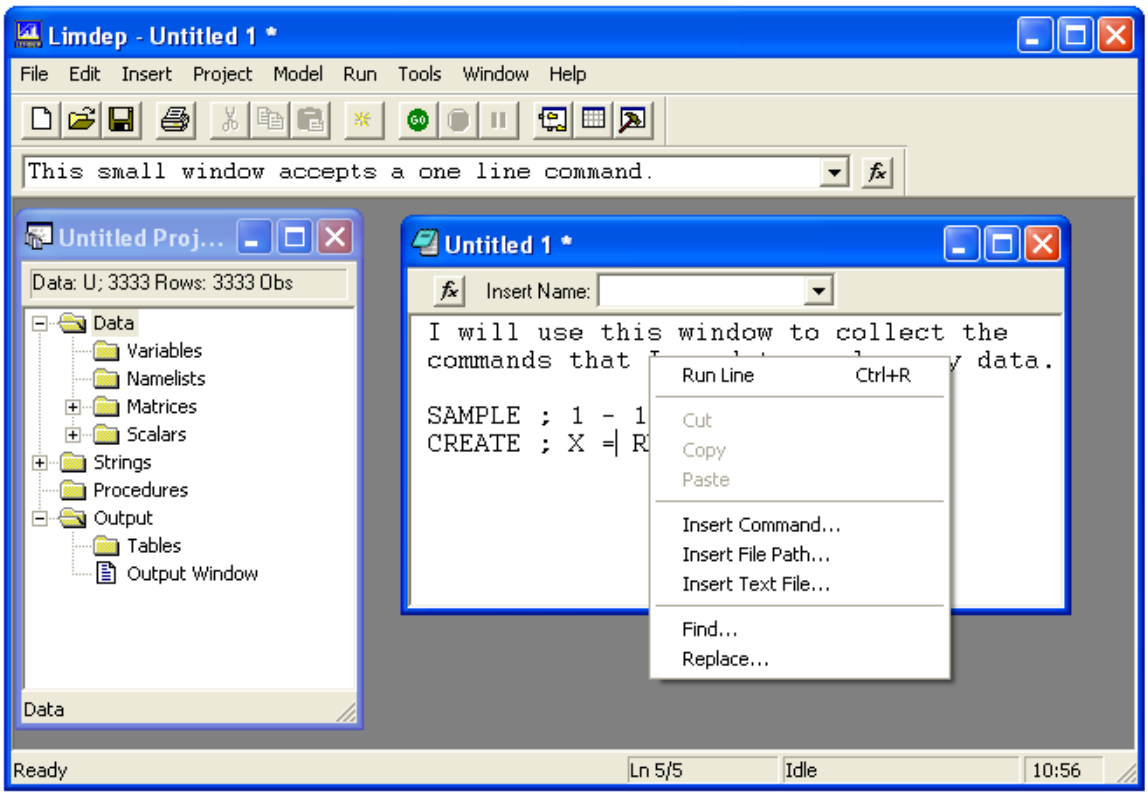


Figure 3.8 The Edit/Right Mouse Button Menu

You may have noted by this point, that operating of *LIMDEP* uses a mixture of menus and commands submitted from the text editor. Though most modern software is largely "point and click" menu driven, much of what you do in econometrics is not very well suited to this style of processing. In fact, it is possible to operate *LIMDEP* almost without using your keyboard once your data are entered and ready. We will demonstrate use of the menus for model building as we go along. But, you will almost surely find that using the command editor and the *LIMDEP* command language is far simpler and more efficient than using the menus and dialog windows. Moreover, when you do begin to write your own computations, such as using matrix algebra where you must compose mathematical expressions, then the menus will no longer be useful.

3.4 A Short Tutorial

Start the Program.

We assume that you have successfully installed *LIMDEP* on your computer and created a shortcut to use to invoke the program. Now, invoke *LIMDEP*, for example, by double clicking the shortcut icon or from the Start:Programs menu. The desktop will appear as shown in Figure 3.9, with a new project window open, and no other windows active.

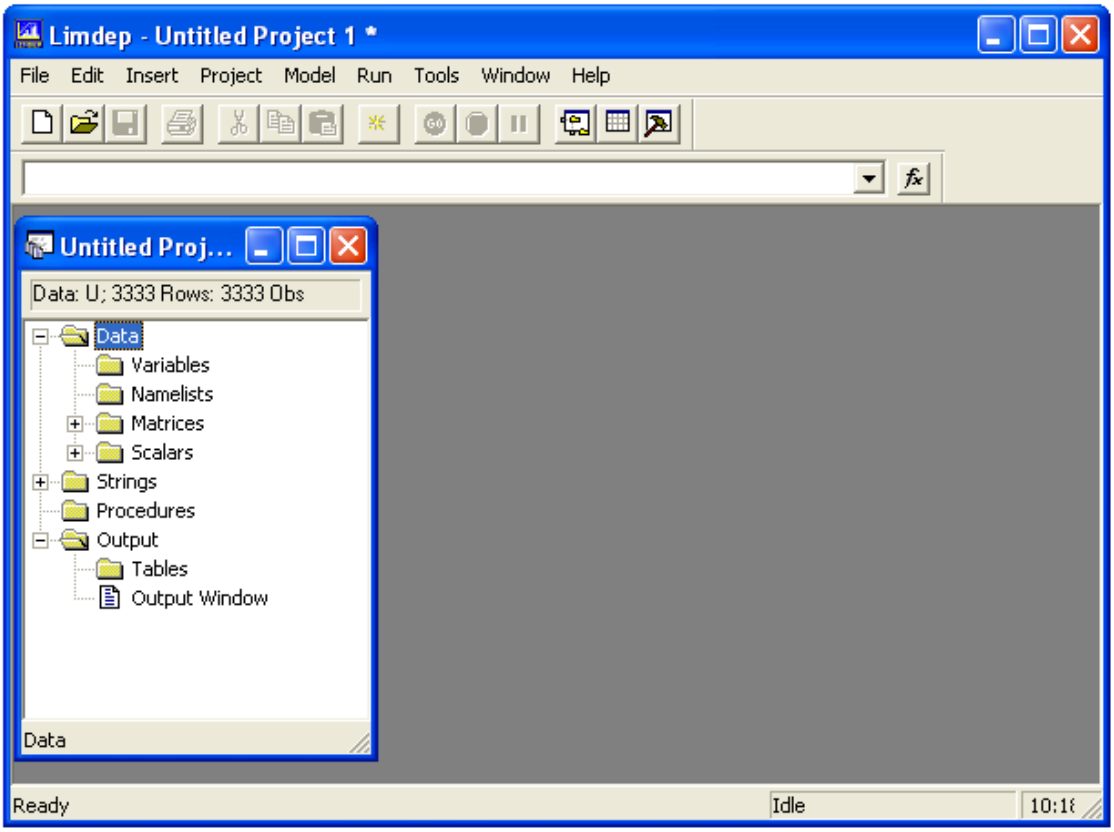


Figure 3.9 Initial *LIMDEP* Desktop

Open an Editing Window.

Select File:New, then Text/Command Document to open an editing window, exactly as discussed in Section 3.2.2. (See Figure 3.3.)

Place Commands in the Editing Window.

Type the commands shown in the editing window of Figure 3.10. 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. (You need not type the comment lines "I will use ... data."

Submit the First Two Commands.

Highlight the first two lines of this command set. Now, move the mouse cursor up to the (now) green button marked GO – it is directly below Tools – and click the GO button. Note that a new window appears, your output window – see Figure 3.11. (You may have to resize it to view the output.)

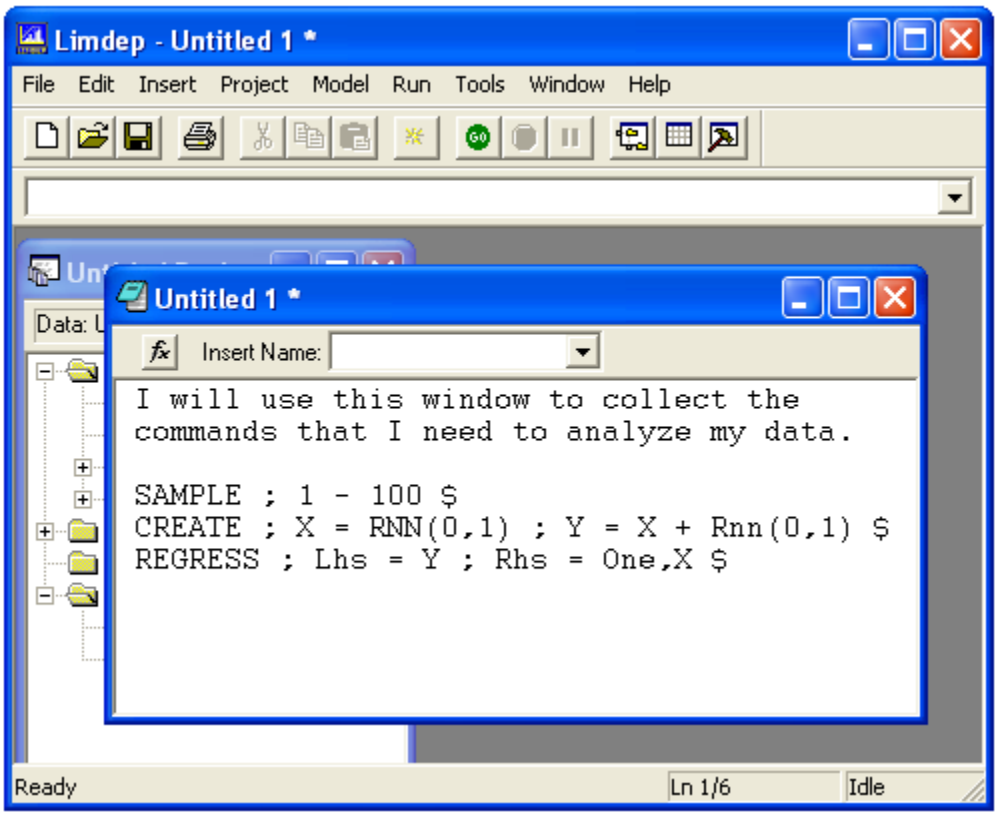


Figure 3.10 Editing Window

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.

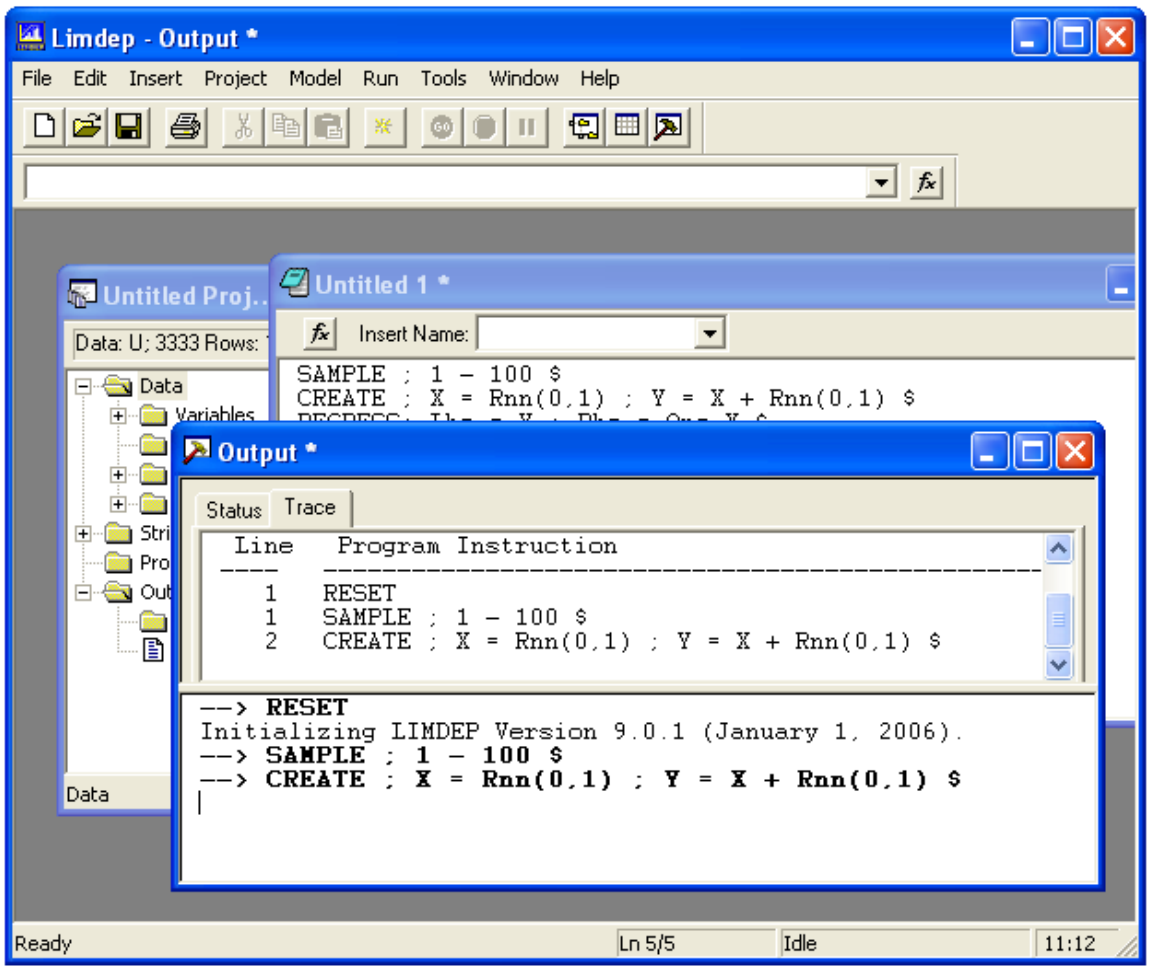


Figure 3.11 Output Window with Command Echo

Compute the Regression.

Before doing this step, 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 trace 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.

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. (Exit status of 0.0 means that the model was estimated successfully.)

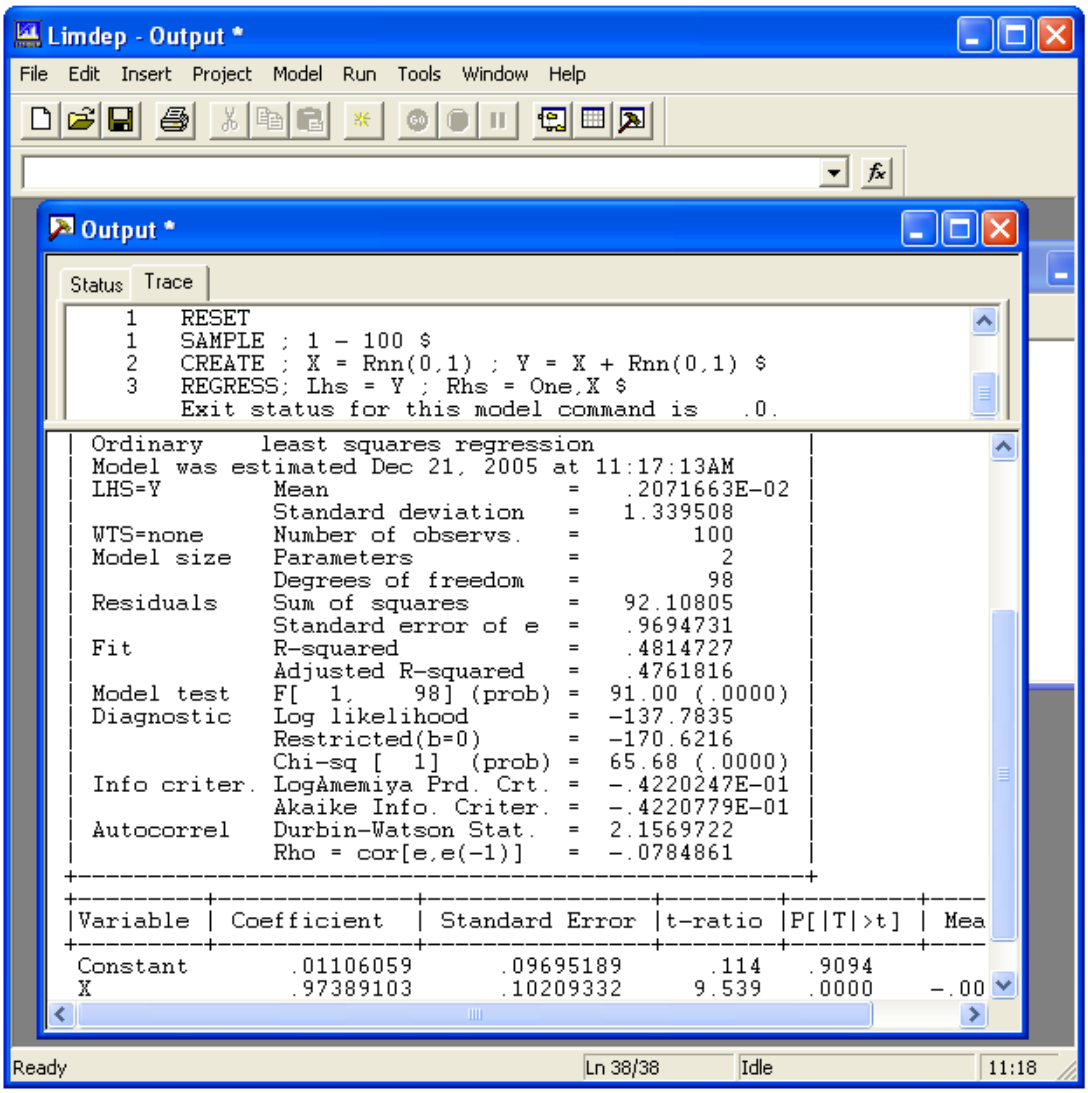


Figure 3.12 Regression Output in Output Window

The Project Window.

Note in Figure 3.9, 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 3.13.

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.

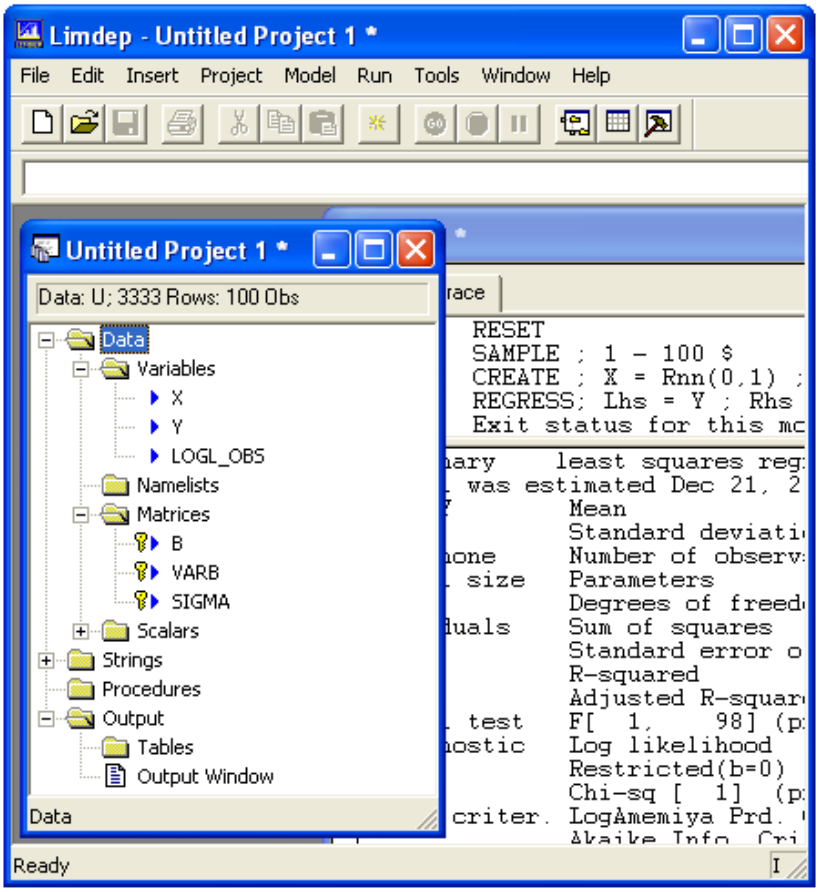


Figure 3.13 Project Window

Exit the Program.

When you are done exploring, select File:Exit. There is no need to save any of these windows, so answer no to the three queries about saving your results.

3.5 Help

NOTE: *LIMDEP*'s help feature uses a .hlp format file. This is (was) a standard Windows format feature, and its operation relies on a Windows utility program. With the release of Windows Vista, without notification, Microsoft made a decision to end support of this format, and did not include a version of the help engine in the new operating system. Thus, the Help procedure in *LIMDEP* (and many other programs) is not compatible with Windows Vista. But, evidently in response to the predictable protest, Microsoft has produced a patch for this feature that can be added to the operating system to revive help engines such as ours. Please go to http://www.limdep.com/support/faq/ for more information about this as well as instructions on how to obtain the patch.

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.' In the first book, you will find a selection of *Topics* that discuss general aspects of operating the program. 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 a summary of the various parts of the desktop. The fourth book contains descriptions of updates to *LIMDEP* that were added after the manual went to press. Finally, there are three books of useful ancillary material: a collection of *LIMDEP* programs, some of which appear in the manual for the program, 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:\LIMDEP9, and there are three subfolders, Data Files, LIMDEP Command Files, and Project Files.

Chapter 4: *LIMDEP* Commands

4.1 Commands

There are numerous menus and dialog boxes provided for giving instructions to *LIMDEP*. 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 chapter will describe the *LIMDEP* command language. We begin by describing the general form and characteristics of *LIMDEP* commands. Section 4.4 will illustrate how the menus and dialog boxes can also be used to operate the program.

4.2 Command Syntax

All program instructions are of the form:

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

The verb is a unique four character name 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 fields separated by semicolons (**;**). Commands always end with a '**$.**' 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) ;  bols = <X'X> * X'y $
REGRESS  ; Lhs = logq ; Rhs = one, Log(k), Log(l) ; Plot residuals $
```

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. But, note that this implies that you cannot use upper and lower case as if they were different in any respect. That is, the variable CAPITAL is the same as capital.
- You may put spaces anywhere in any command. *LIMDEP* will ignore all spaces and tabs in any command.
- Every command must begin on a new line in your text editor
- 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.

4.3 Naming Conventions and Reserved Names

Most commands refer to entities such as variables, groups of variables, matrices, procedures, and particular scalars by name. Note in our examples, we have referred to 'x,' '*y*,' *abd* '*capital*.' 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 4.1 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 that would arise if you try to name a variable or a matrix with 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.

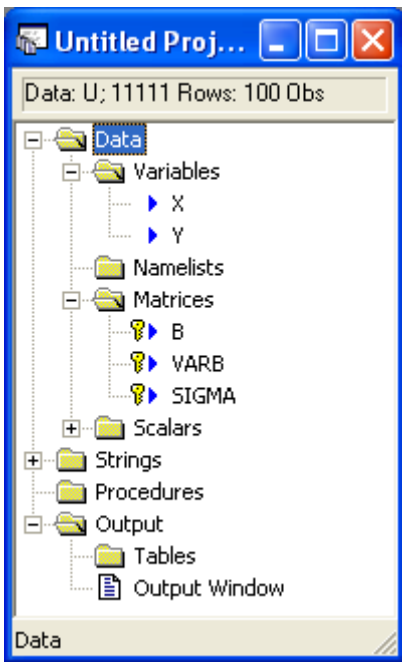


Figure 4.1 Reserved Names

4.4 Command Builders

The desktop provides menus and dialog boxes for the functions that you need to manage your data, open projects and text windows, and so on. In almost most all cases, there are commands that you can use for these same functions, though generally you will use the menus. The reverse is true when you analyze your data and do statistical and econometric analysis. You will usually use commands for these procedures. However, we note, for most statistical operations, there are also menus and dialog boxes if you prefer to use them.

For an example, in Figure 4.2, you can see an editing window that contains four commands, a SAMPLE command, followed by two CREATE commands that create a simulated data set on 'X; and 'Y.' and a regression command. The first three commands have already been executed, and the data, as you can see in the project window, are already created. I can now compute the regression by highlighting the REGRESS command and pressing the GO button.

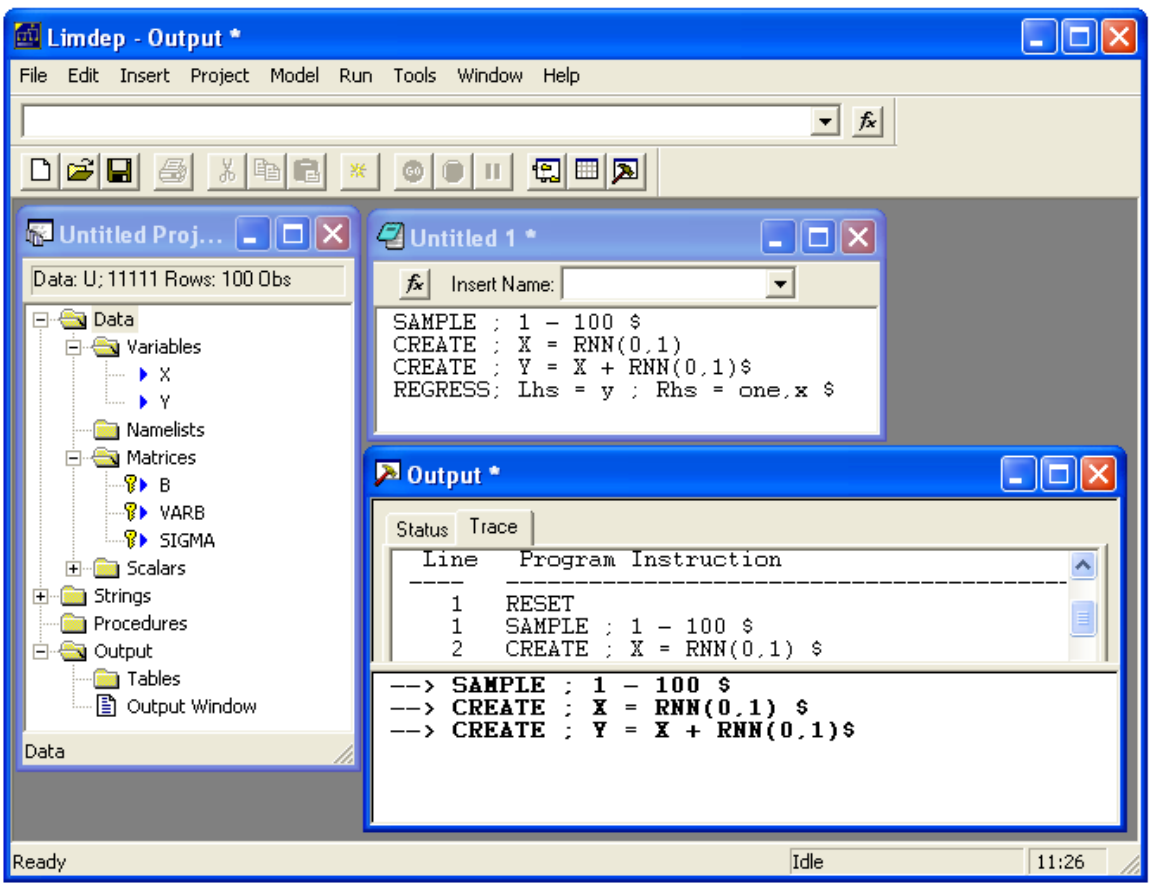


Figure 4.2 Regression Command

An alternative way to request the regression is to use the Model menu on the desktop and locate the command builder for the linear regression. This is shown in Figure 4.3. When I select this item from the menu, this will open a dialog box that lets me construct my linear regression model without typing the commands in the editor. The command builder is shown in Figure 4.4.

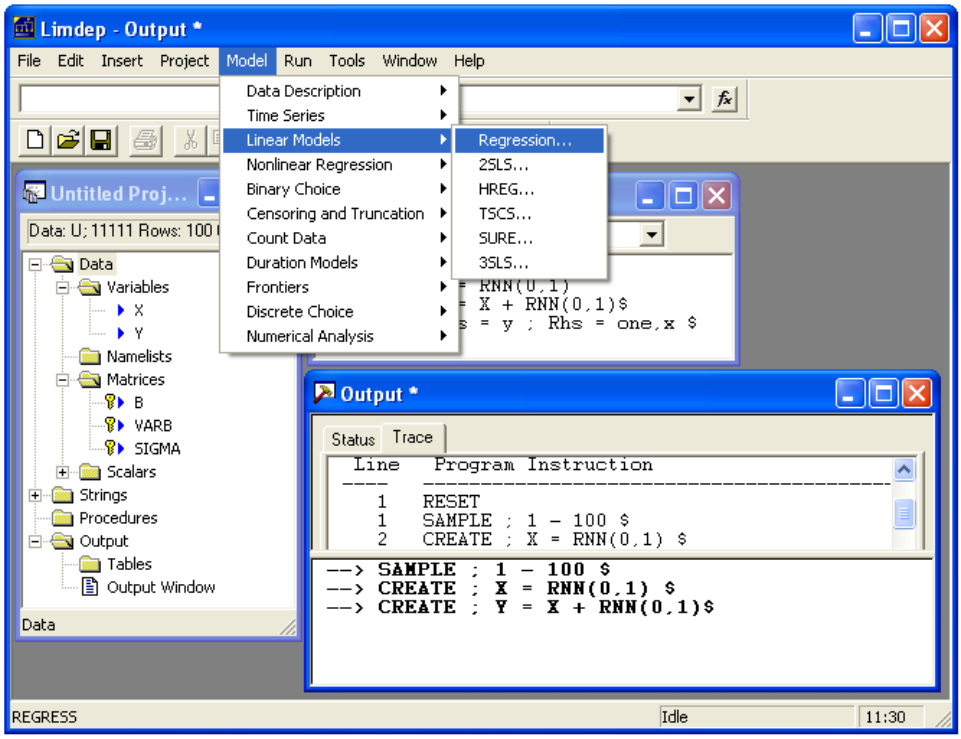


Figure 4.3 Model Menu

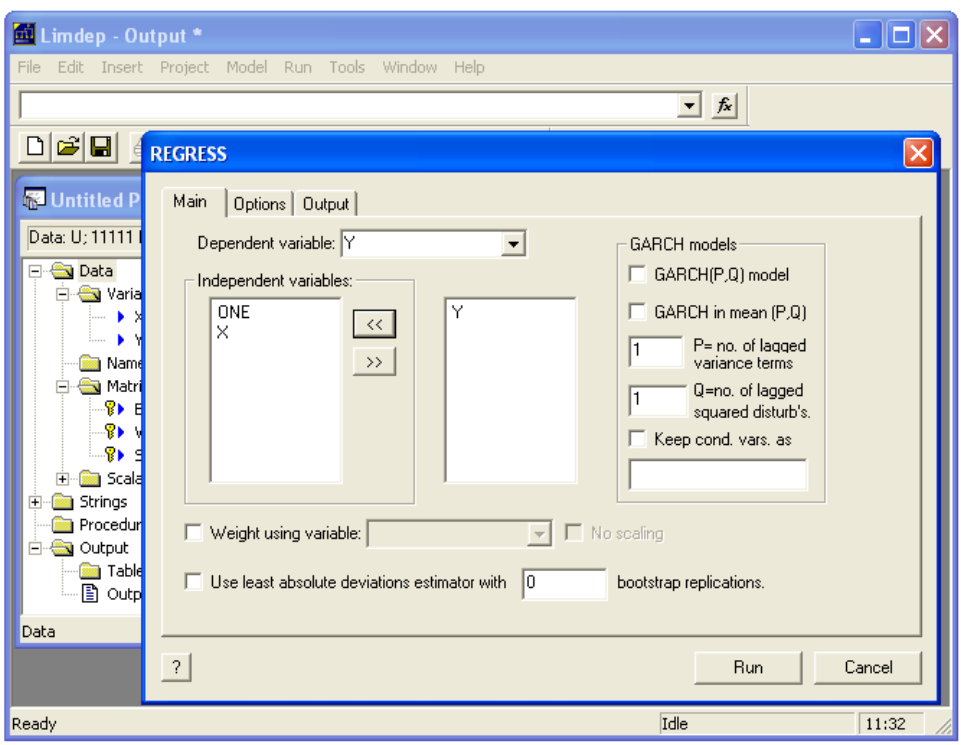


Figure 4.4 Regression Command Builder

After selecting the variables that I want to appear on the left and right hand sides of my regression equation, I press Run, and the regression is computed. The results are shown in Figure 4.5.

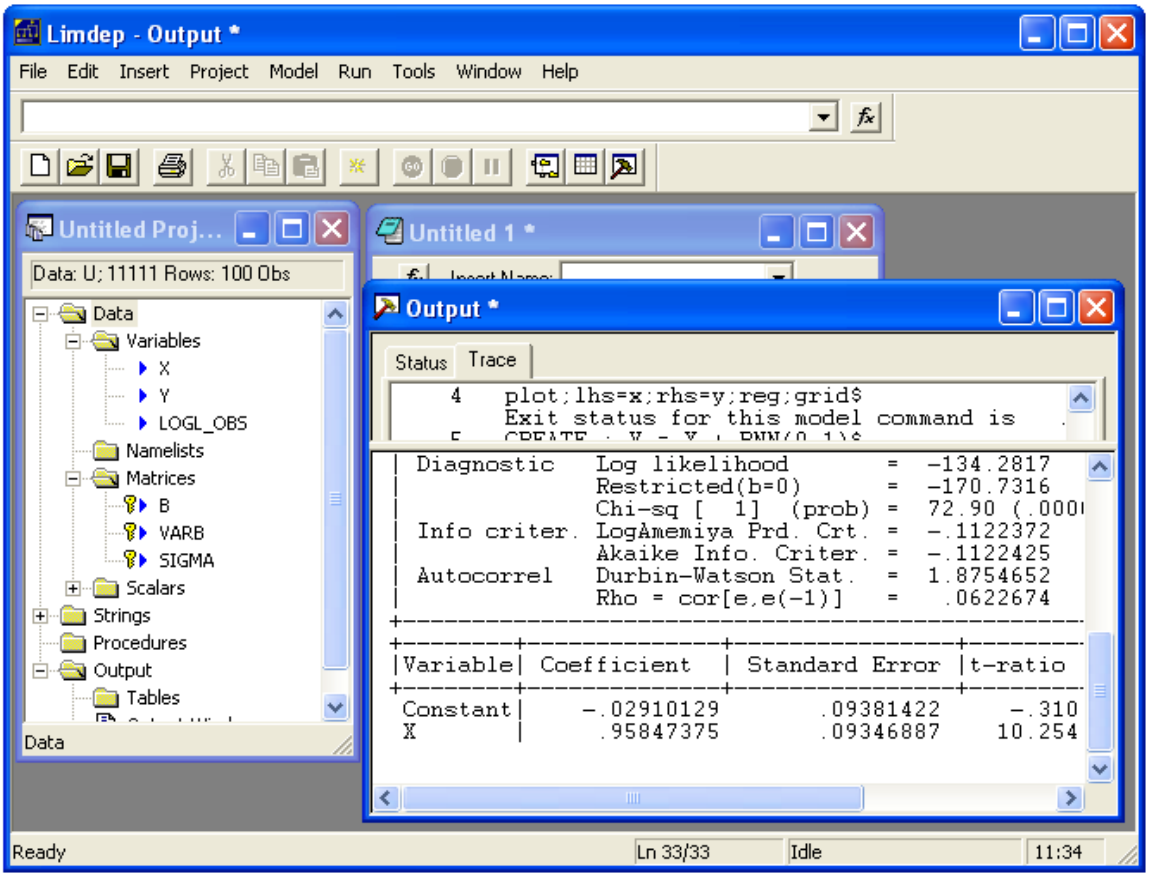


Figure 4.5 Regression Results in Output Window

The dialog box just shown is called a 'Command Builder' because in addition to submitting the necessary information to the program to carry out the regression, it literally creates the command you have issued, and places it in the output window. You can see this result in Figure 14.6, where the '**REGRESS**' command is echoed in the output window before the results. You can edit/copy and paste this command to move it to your editing window where you may change it and use it as if you had typed it yourself.

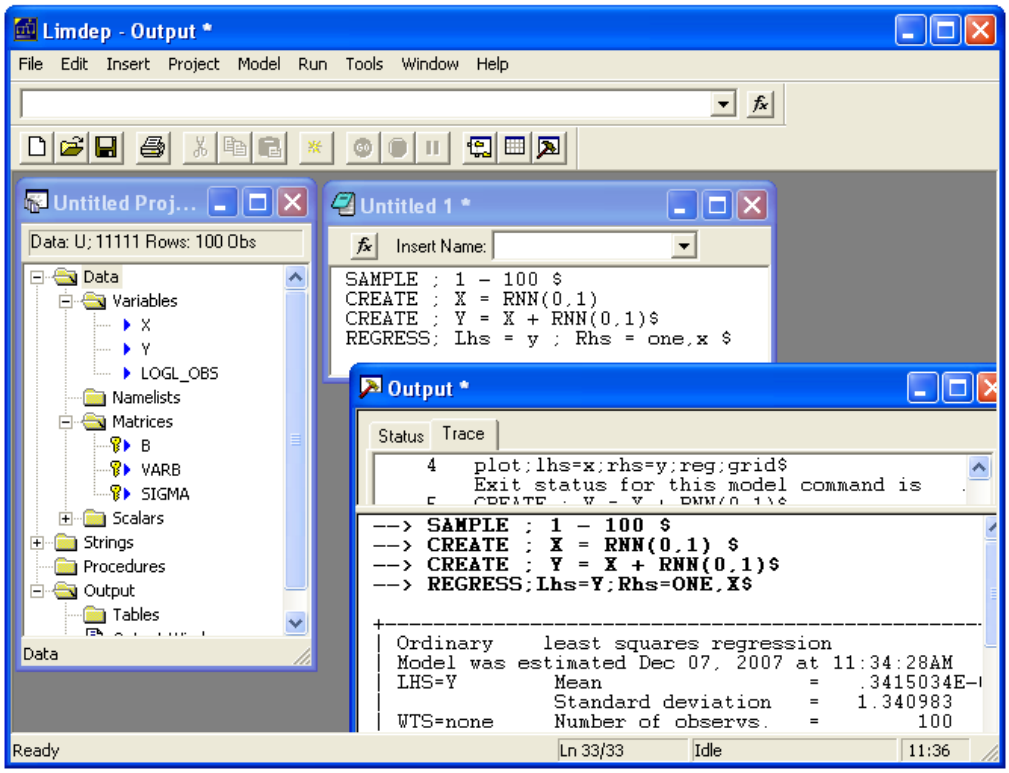


Figure 4.6 Command Builder Output

Chapter 5: Program Output

5.1 The Output Window

LIMDEP will automatically open an output window and use it for the display of results produced by your commands. Figure 5.1 below 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.

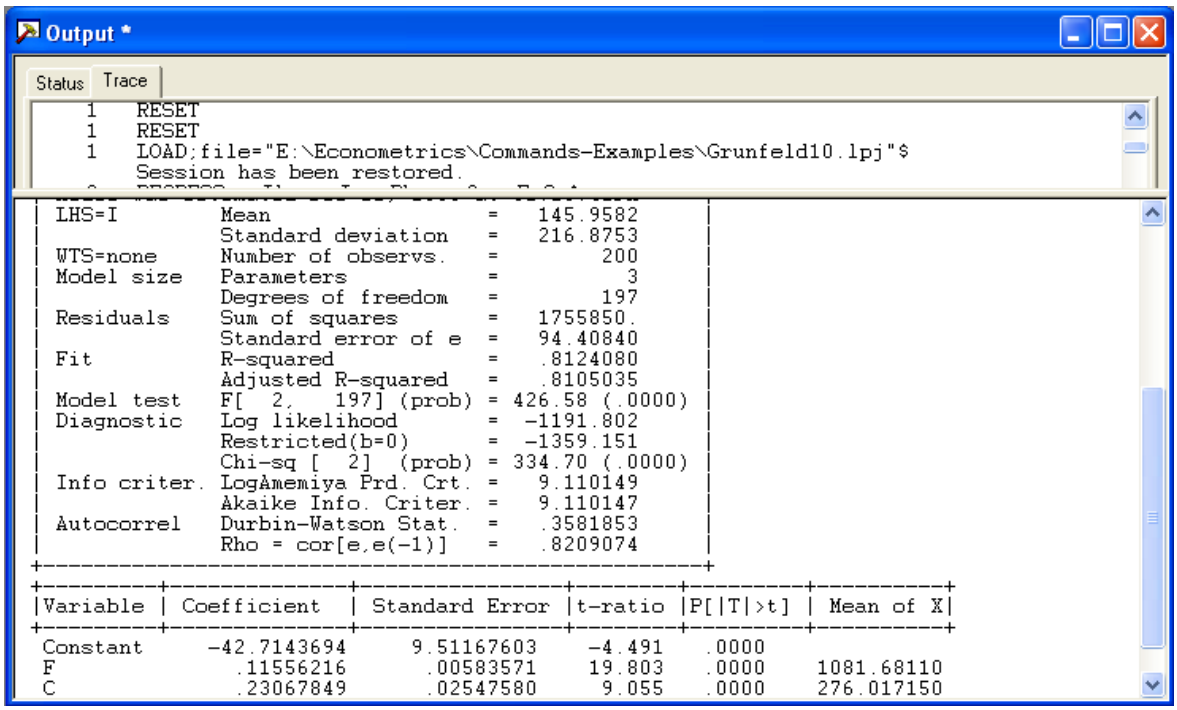


Figure 5.1 Output Window

5.2 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.

But, there is a way to get full editing capability. You can select, then cut or copy the 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.

5.3 Exporting Your Output.

When you wish to use your statistical results in a report, you will generally copy the results from your output window directly into your word processor. Results in the window are displayed in the default 9 point Courier New font. (You can change this to any font you wish using Tools:Options/Editor.) All numerical results are shown in this form. Graphical results may also be copied directly off the screen. They are produced in a ".wmf" (windows metafile) format. You can paste a graph into your word processor, then resize the figure to whatever size you like. An example appears below in Figure 5.2. An extensive application appears in the next chapter.

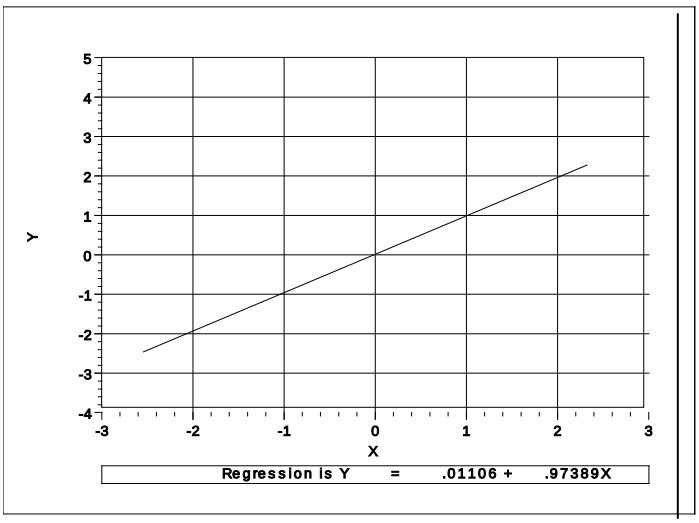


Figure 5.2 Figure Exported From LIMDEP Output

Chapter 6: Application and Tutorial

6.1 Application – An Econometrics Problem Set

This chapter will illustrate most of the features in *LIMDEP* that you will use for basic econometric analysis. We will use a simulated problem set to motivate the computations. This application will use many of the commands and computational tools in the program. Of course, we have not documented these yet – they appear later in the manual in the succeeding chapters. We do anticipate, however, that you will be able to proceed from this chapter alone to further use of the program.

6.2 Assignment: The Linear Regression Model

The data listed below are a set of yearly time series observations (1953-2004) on the U.S. Gasoline Market. Use these data to perform the following analyses:

1. Read the raw data into *LIMDEP*.

2. Obtain descriptive statistics (means, standard deviations, etc.) for the raw data in your data set.

3. The data are in levels. We wish to fit a constant elasticity model, which will require that the variables be in logarithms. Obtain the following variables:

   logG = log of per capital gasoline consumption
   logPg = log of the price (index) of gasoline
   logI = log of per capita income
   logPnc = log of price (index) for new cars
   logPuc = log of price (index) for used cars
   logPpt = log of price (index) for public transportation
   t = time trend = *year* – 1952.

4. We notice immediately that if we intend to use the (logs of the) price indexes in our regression model, there is at least the potential for a problem of multicollinearity. As a preliminary indication of how serious the problem is likely to be, obtain a time series plot of the four price series.

5. We begin with a simple 'demand' equation. Obtain the least squares regression results in a regression of *logG* on the log of the price index, *logPg*. Report your regression results

   a. Notice that the coefficient on log price is positive. Shouldn't a demand curve slope downward? What is wrong here?

   b. Now, add the obviously missing income variable to the equation. Compute the linear regression of *logG* on *logI* and *logPg*. Report your results and comment on your findings.

6. The full regression model that you will explore for the rest of this exercise is

    a. $logG = \beta_1 + \beta_2 logPg + \beta_3 logI + \beta_4 logPnc + \beta_5 logPuc + \beta_6 logPpt + \beta_7 t + \varepsilon$

7. Obtain the least squares estimates of the coefficients of the model. Report your results. Obtain a plot of the residuals as part of your analysis.

    a. (For more advanced courses) Compute the least squares regression coefficients, **b**, s^2, the covariance matrix for **b**, and R^2 using matrix algebra.

    b. Test the hypothesis that all of the coefficients in the model save for the constant term are zero using an F test. Obtain the sample *F* and the appropriate critical value for the test. Use the 95% significance level.

8. Still using the full model, use an *F* test to test the hypothesis that the coefficients on the three extra prices are all zero

9. Test the hypothesis that the elasticities with respect to the prices of new and used cars are equal.

    a. Do the test using an *F* test.

    b. Use a *t* test, showing all computations.

10. The market for fossil fuels changed dramatically at the end of 1973. We will examine whether the data contain clear evidence of this phenomenon.

    a. Use a Chow test for structural break to ascertain whether the same model applies for the period 1953 – 1973 as for 1974 – 2004.

    b. (More advanced) Use a Wald test to test the hypothesis. How do the assumptions underlying these two tests differ?

In the analysis to follow, we will work through the problem set in detail. You can work through the steps with the presentation by typing the commands into the text editor. To save time typing, you can just upload the file Tutorial.lim that you will find in the

C:\LIMDEP9\LIMDEP Command Files

folder. The data set that we will use is also provided as file Gasoline.txt and Gasoline.xls in the folder

C:\LIMDEP9\LIMDEP Data Files

The data for the exercise are listed below.

Year	GasExp	GasPrice	Income	PNewCar	PUsedCar	PPubTrn	Pop
1953	7.4	16.668	8883	47.2	26.7	16.8	159565
1954	7.8	17.029	8685	46.5	22.7	18.0	162391
1955	8.6	17.210	9137	44.8	21.5	18.5	165275
1956	9.4	17.729	9436	46.1	20.7	19.2	168221
1957	10.2	18.497	9534	48.5	23.2	19.9	171274
1958	10.6	18.316	9343	50.0	24.0	20.9	174141
1959	11.3	18.576	9738	52.2	26.8	21.5	177130
1960	12.0	19.112	9770	51.5	25.0	22.2	180760
1961	12.0	18.924	9843	51.5	26.0	23.2	183742
1962	12.6	19.043	10226	51.3	28.4	24.0	186590
1963	13.0	18.997	10398	51.0	28.7	24.3	189300
1964	13.6	18.873	11051	50.9	30.0	24.7	191927
1965	14.8	19.587	11430	49.7	29.8	25.2	194347
1966	16.0	20.038	11981	48.8	29.0	26.1	196599
1967	17.1	20.700	12418	49.3	29.9	27.4	198752
1968	18.6	21.005	12932	50.7	30.7	28.7	200745
1969	20.5	21.696	13060	51.5	30.9	30.9	202736
1970	21.9	21.890	13567	53.0	31.2	35.2	205089
1971	23.2	22.050	14008	55.2	33.0	37.8	207692
1972	24.4	22.336	14270	54.7	33.1	39.3	209924
1973	28.1	24.473	15309	54.8	35.2	39.7	211939
1974	36.1	33.059	15074	57.9	36.7	40.6	213898
1975	39.7	35.278	15555	62.9	43.8	43.5	215981
1976	43.0	36.777	15693	66.9	50.3	47.8	218086
1977	46.9	38.907	15991	70.4	54.7	50.0	220289
1978	50.1	40.597	16674	75.8	55.8	51.5	222629
1979	66.2	54.406	16843	81.8	60.2	54.9	225106
1980	86.7	75.509	16711	88.4	62.3	69.0	227726
1981	97.9	84.018	17046	93.7	76.9	85.6	230008
1982	94.1	79.768	17429	97.4	88.8	94.9	232218
1983	93.1	77.160	17659	99.9	98.7	99.5	234333
1984	94.6	76.005	18922	102.8	112.5	105.7	236394
1985	97.2	76.619	19622	106.1	113.7	110.5	238506
1986	80.1	60.175	19944	110.6	108.8	117.0	240683
1987	85.4	62.488	19802	114.6	113.1	121.1	242843
1988	88.3	63.017	20682	116.9	118.0	123.3	245061
1989	98.6	68.837	21048	119.2	120.4	129.5	247387
1990	111.2	78.385	21379	121.0	117.6	142.6	250181
1991	108.5	77.338	21129	125.3	118.1	148.9	253530
1992	112.4	77.040	21505	128.4	123.2	151.4	256922
1993	114.1	76.257	21515	131.5	133.9	167.0	260282
1994	116.2	76.614	21797	136.0	141.7	172.0	263455
1995	120.2	77.826	22100	139.0	156.5	175.9	266588
1996	130.4	82.596	22506	141.4	157.0	181.9	269714
1997	134.4	82.579	22944	141.7	151.1	186.7	272958
1998	122.4	71.874	24079	140.7	150.6	190.3	276154
1999	137.9	78.207	24464	139.6	152.0	197.7	279328
2000	175.7	100.000	25380	139.6	155.8	209.6	282429
2001	171.6	96.289	25449	138.9	158.7	210.6	285366
2002	163.4	90.405	26352	137.3	152.0	207.4	288217
2003	191.3	105.154	26437	134.7	142.9	209.3	291073
2004	224.5	123.901	27113	133.9	133.3	209.1	293951

6.3 Read the Raw Data

In order to analyze your data, you must "read" them into the program. There are many ways to read your data into *LIMDEP*. We will consider two simple ones, via an Excel spreadsheet and as portable ASCII text. Others are described in the full manual for the software.

Excel Spreadsheet

If your data are given to you in the form of an Excel spreadsheet, then you can "import" them directly into the program. The first step is to use the Project menu on the desktop as shown in Figure 6.1. This would then launch a mini-explorer. You would then locate your file in the menu. Finally, you can just double click the file name and the data are read into the program.

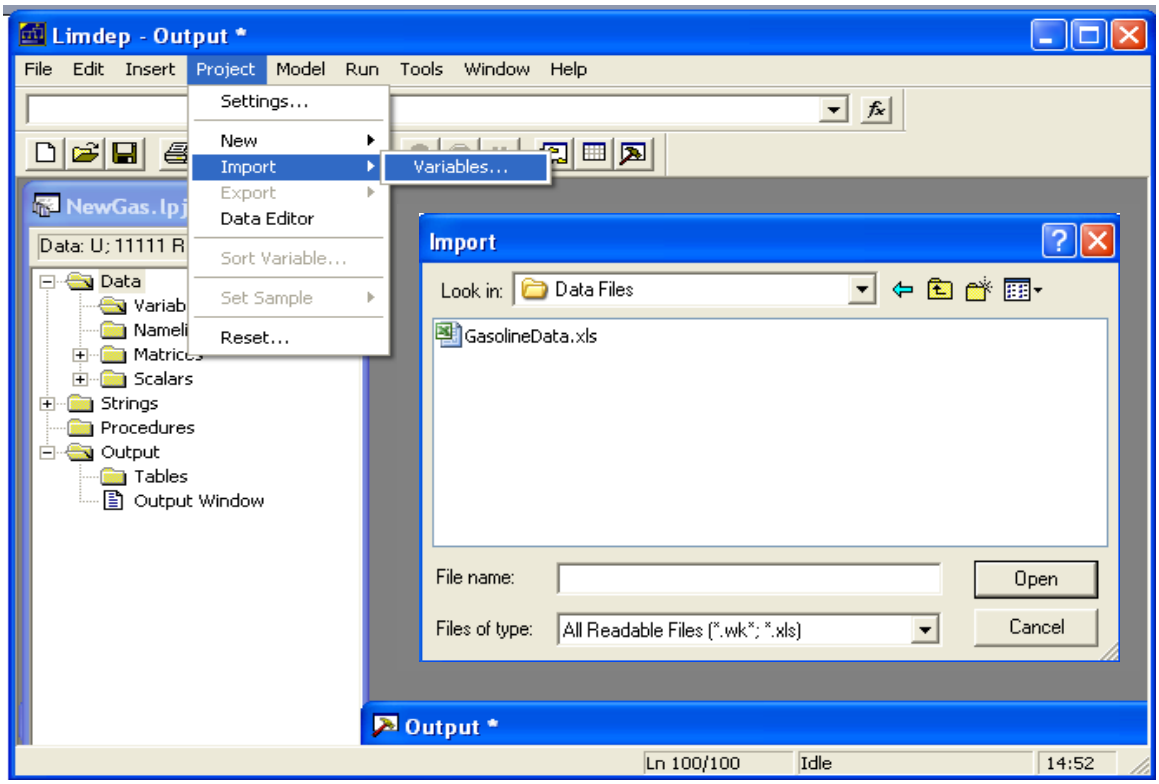


Figure 6.1 Reading an Excel Spreadsheet File

WARNING: For at least 15 years, until 2007, the .XLS format has served as a lingua franca for exchanging data files between programs. Almost any statistical package could read one. Beginning in 2007, Microsoft drastically changed the internal format of spreadsheet files, and *Excel 2007* files are no longer compatible with other software. It will be quite a while (if ever) for software authors to catch up with this change, so for the present, we note that *LIMDEP* cannot read an *Excel 2007* spreadsheet file. If you are using *Excel 2007*, be sure to save your spreadsheets in the 2003 format.

Text Input File

A second, equally simple way to read your data is simply to read them off the screen in a text editor. The file GasolineData.lim contains text that is exactly the highlighted command, names and data that appear in Figure 6.3. To get these data into the program, we can proceed as follows: First, we start *LIMDEP*. Then, use File:Open and navigate to the file. You will reach the file in the mini-explorer.

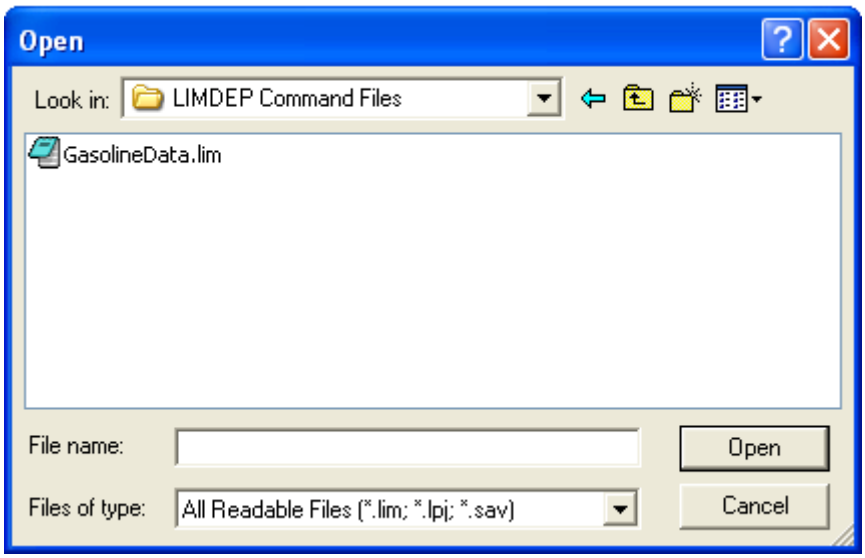


Figure 6.2 Mini-explorer for Data File

Now, just double click the file name, and the contents of the file will be placed in a new text editing window as shown in Figure 6.3. With the data in this form, all that is needed now is Edit/Select All, then click the GO button,  in the desktop toolbar, and the data will be read from the screen.

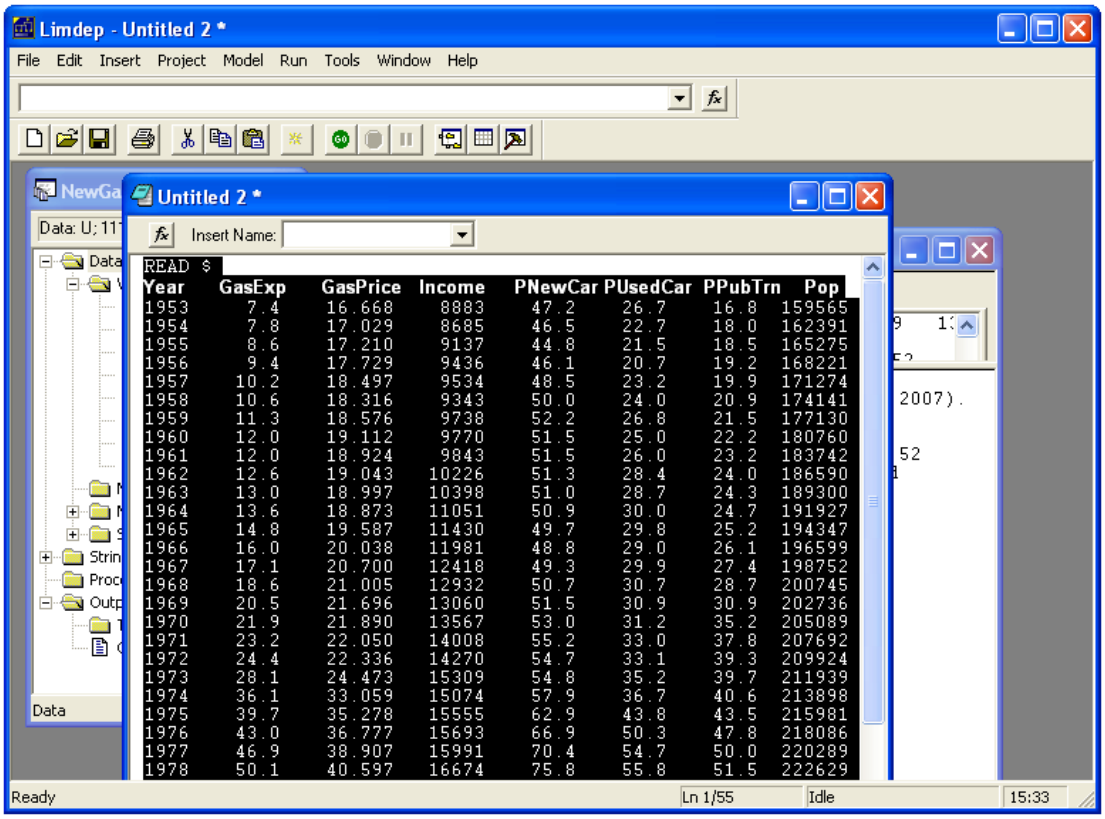


Figure 6.3 Reading ASCII Data in a Text Editor

Using Copy/Paste

For purposes of this method, suppose you have been given the data table in a document file that will allow you to copy and paste the data file exactly the way they appear on the page. Figure 6.4 below shows the example for our tutorial. The data that we wish to analyze appear on the left hand page. To transport these data into *LIMDEP*, we could proceed as follows: First, start *LIMDEP*. Then, we open a text editor in *LIMDEP* with File:New/Text/Command Document Type the command "**READ$**" in the top line. (Do not forget the **$** at the end of the command.) Finally, just highlight the data (and names line) in the document, use Edit/Copy in the word processor and Edit/Paste in *LIMDEP*.

The end result of either of these approaches will be to place the data in the text editor, ready for you to use.

Running the Data File as a Command Set

Placing the **READ$** command and data in the text editor, then highlighting the material and clicking GO to execute the **READ$** actually involves an extra step. You can instruct *LIMDEP* to do the whole thing at once, since the file GasolineData.lim contains the command and the data. If you use Run:Run File from the desktop menu, then select the .lim file from the mini-explorer, *LIMDEP* will read the file and internally, by itself, highlight the material and press its own GO button. This process will carry out the **READ$** command and proceed directly to the confirmation shown in Figure 6.5.

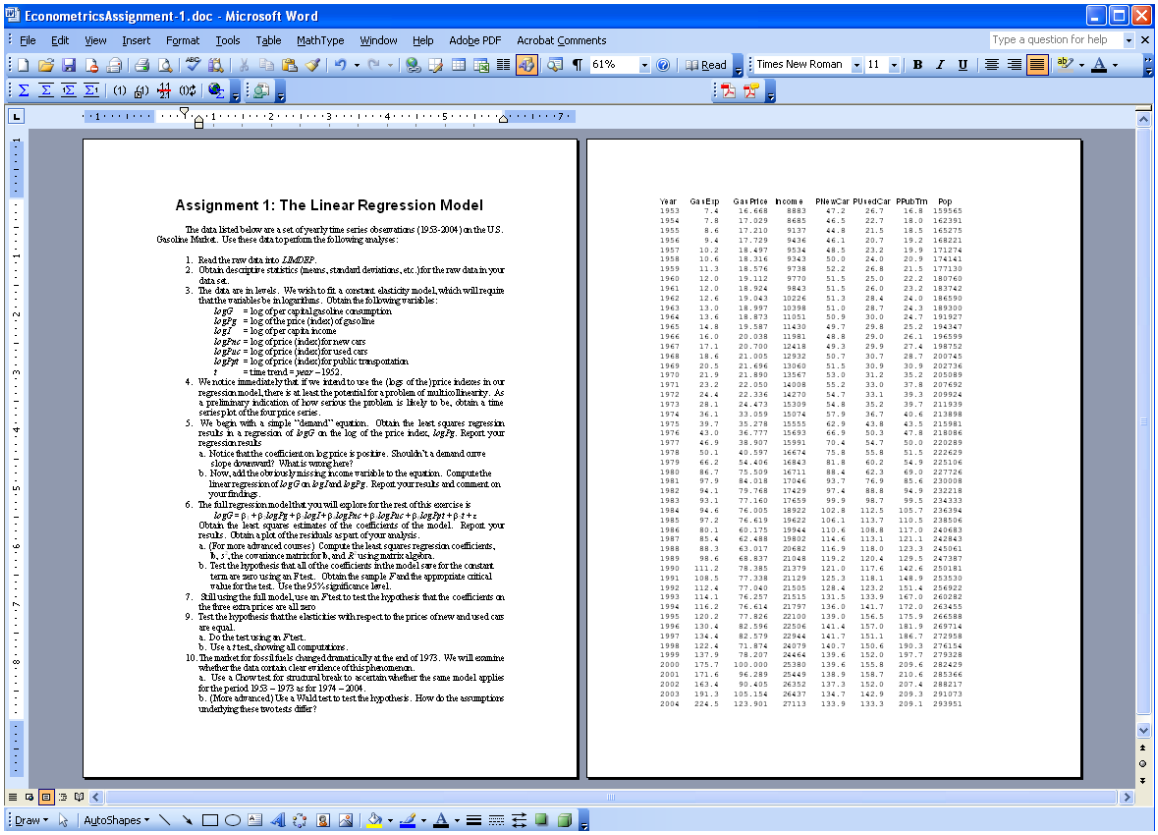


Figure 6.4 Word Processor File Containing the Data Set

The confirmation in the output window and the appearance of the names of the variables in the project window indicate that the raw data have been read and we can now analyze them.

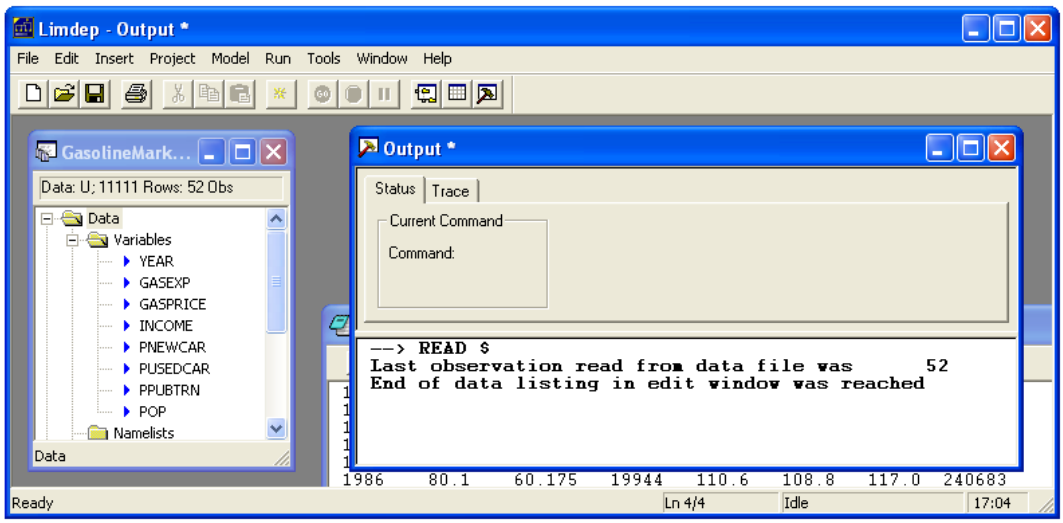


Figure 6.5 Data Confirmation

You can verify that the data have been properly imported by looking at the data editor. This is a small spreadsheet editor. You might use it to enter a small data set with a few observations by typing them directly into the program's memory. The data editor will display the data that have already been placed in memory. To open the data editor, click the Data Editor button on the desktop toolbar, as shown in Figure 6.6.

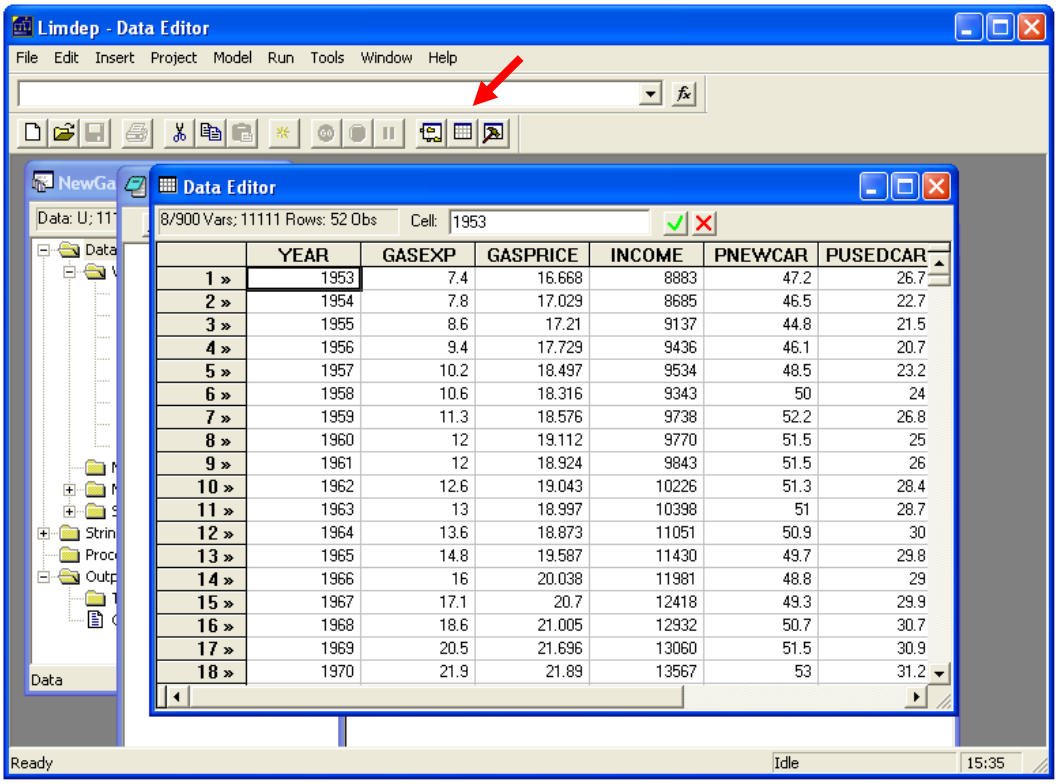


Figure 6.6 Data Editor

Rereading the Data

Though each one of these methods of reading the data into the program is short and simple, it might still seem like a bit of effort just to get the data ready to use. The first time you use a data set, there has to be some way to get the data ready for this program to use them. You could just type them in yourself – there is a spreadsheet style editor as shown in Figure 6.6. But, if the data already exist somewhere else, that would be terribly inefficient and inconvenient. We do emphasize, however, that however you enter your data set into *LIMDEP* the first time, you will only do it once. Once the data are in the program and in a project, you will save that project in a file. Thereafter, when you want to use these data again, you will simply double click the project, in a mini-explorer, on your desktop, in *LIMDEP*'s recently used files, or somewhere else, and you will be ready to go back to work. See Section 6.5.3 for further discussion.

6.4 Tutorial Commands

In this tutorial, you will used nearly all of the features of *LIMDEP* that you will need to complete a graduate course in econometrics. We will run through a long sequence of operations one at a time, and display the results at each step. You will want to carry out these operations as we do. Rather than type them one at a time, it will be more convenient for you to have them available in a text editor, where you can execute each one just by highlighting one or more lines and clicking the GO button,  .

The listing below contains all of the commands that we used in this tutorial. You can place them in your screen editor by using File:Open then making your way to

C:\LIMDEP9\LIMDEP Command Files\Tutorial.lim

Select this file. Your screen will then contain all the commands that will be used in the tutorial to follow. (The text editor does not replicate the boldface in the listing below.) You can then 'follow along' by highlighting and executing each command or set of commands as they are discussed below.

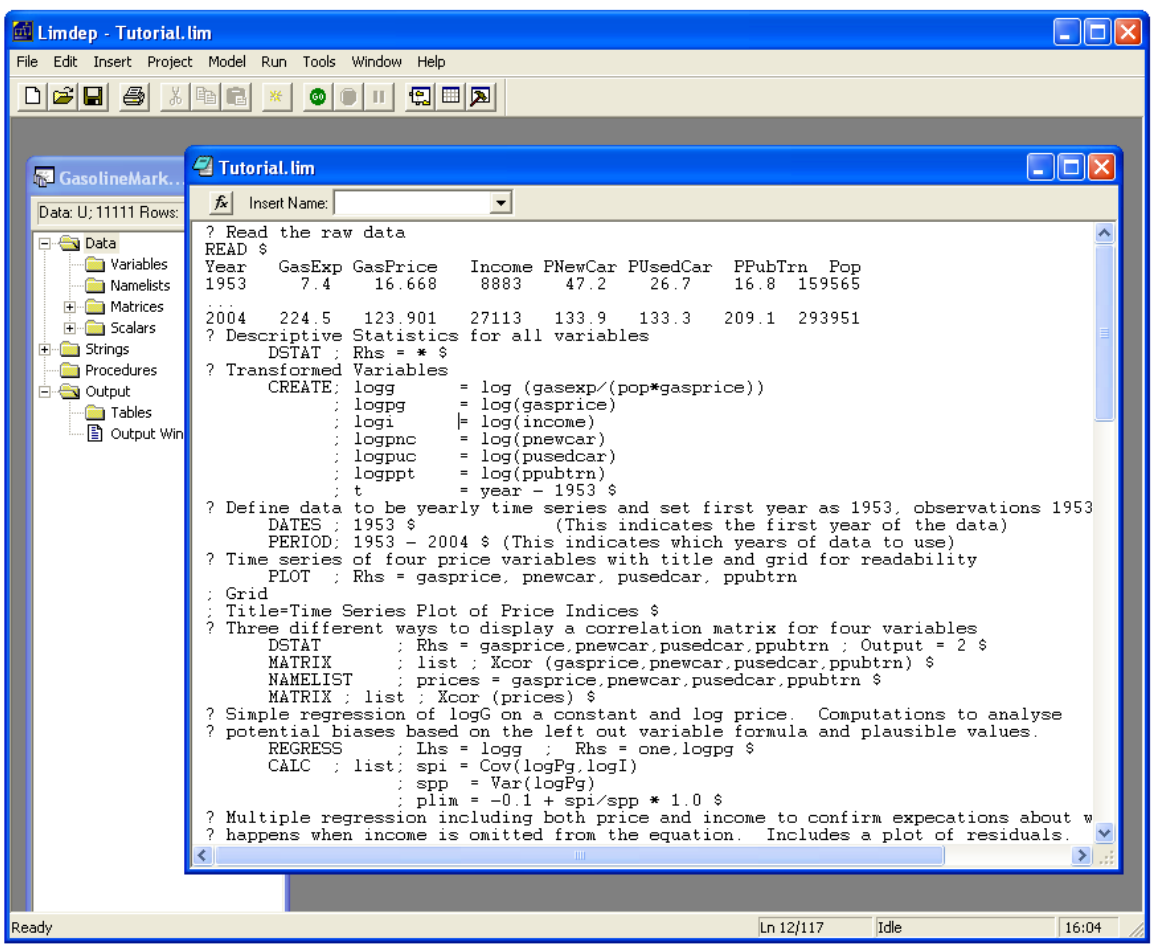


Figure 6.7 Tutorial Commands in Text Editor

These are the commands contained in the tutorial command file. Lines that begin with a question mark are comments. If you submit these lines to the program as if they were commands, they are ignored.

```
? Read the raw data
        READ $
Year    GasExp GasPrice    Income PNewCar PUsedCar  PPubTrn  Pop
1953      7.4    16.668     8883    47.2    26.7     16.8  159565
...
2004    224.5   123.901   27113   133.9   133.3    209.1  293951
? Descriptive Statistics for all variables
        DSTAT       ; Rhs = * $
? Transformed Variables
        CREATE      ; logg    = log (gasexp/(pop*gasprice))
                    ; logpg   = log(gasprice)
                    ; logi    = log(income)
                    ; logpnc = log(pnewcar)
                    ; logpuc = log(pusedcar)
                    ; logppt = log(ppubtrn)
                    ; t       = year – 1953 $
? Define data to be yearly time series and set first year as 1953, observations 1953 to 2004.
        DATES       ; 1953 $        (This indicates the first year of the data)
        PERIOD      ; 1953 – 2004 $ (This indicates which years of data to use)
? Time series of four price variables with title and grid for readability
        PLOT        ; Rhs = gasprice, pnewcar, pusedcar, ppubtrn
                    ; Grid
                    ; Title=Time Series Plot of Price Indices $
? Three different ways to display a correlation matrix for four variables
        DSTAT       ; Rhs = gasprice,pnewcar,pusedcar,ppubtrn ; Output = 2 $
        MATRIX      ; list ; Xcor (gasprice,pnewcar,pusedcar,ppubtrn) $
        NAMELIST    ; prices = gasprice,pnewcar,pusedcar,ppubtrn $
        MATRIX ; list ; Xcor (prices) $
? Simple regression of logG on a constant and log price.  Computations to analyze
? potential biases based on the left out variable formula and plausible values.
        REGRESS     ; Lhs = logg  ;  Rhs = one,logpg $
        CALC  ; list    ; spi = Cov(logPg,logI)
                    ; spp  = Var(logPg)
                    ; plim = -0.1 + spi/spp * 1.0 $
? Multiple regression including both price and income to confirm expectations about what
? happens when income is omitted from the equation.  Includes a plot of residuals.
        REGRESS     ; Lhs = logG ; Rhs = one,logpg,logi  ; Plot residuals $
? Full regression model including all variables
        NAMELIST    ; x1 = logpg,logI
                    ; x2 = logpnc,logpuc,logppt
                    ; x =one, x1,x2,t $
        REGRESS     ; Lhs = logg ; Rhs = x ; Plot residuals $
? Least squares computations using matrix algebra, slopes, standard error, R squared,
? covariance matrix for coefficients.  Results are displayed in a table.
```

```
MATRIX      ; bols = <X'X> * X  logg $
MATRIX      ; e  = logg - X * bols $
CALC        ; list  ; se  =  sqr((e'e) / (n-col(X))) ; r2 = 1  - e'e/((n-1)*var(logg))$
MATRIX      ; v = s2 * <X'X>   ; Stat (bols,v,x) $
? F statistic for the hypothesis that all coefficients are zero. R squared for the regression
CALC  ; list    ; f = (r2/(col(X)-1)) / ((1-r2)/(n-col(X))) $
CALC  ; list    ; se  =  sqr(sumsqdev /degfrdm)
                ; r2 = 1  - e'e/((n-1)*var(logg)) $
? F statistic using results retained by the regression.  Critical values for F and t.
CALC  ; list    ; f    = rsqrd/(k-1)/((1-rsqrd)/degfrdm) $
CALC  ; list    ; fc  = Ftb(.95,(kreg-1),degfrdm)
                ; tc  =  ttb(.95, (n-col(x))) $
? Constrained least squares, three coefficients constrained to equal zero.  Test of
? hypothesis using F statistic.  Two ways: Built in and using the R squareds.
REGRESS     ; Lhs = logg ; Rhs = x  ; Cls: b(4) = 0, b(5) = 0, b(6) = 0 $
NAMELIST    ; xu = x  ; xr = one,x1,t $
REGRESS     ; Lhs = logg ; Rhs = xu $
CALC        ; rsqu = rsqrd $
REGRESS     ; Lhs = logg ; Rhs = xr $
CALC        ; rsqr = rsqrd
        ; list  ; f = ((rsqu – rsqr)/Col(x2))/((1-rsqu)/(n-Col(x))) $
? Use the built in calculator function to compute R squared
CALC        ; rsqu = rsq(Xu,logg)
                ; rsqr = rsq(Xr,logg)
        ; list  ; f = ((rsqu – rsqr)/Col(x2))/((1-rsqu)/(n-Col(x))) $
? Testing the hypothesis using the Wald statistic.  Built in command then using matrix algebra.
REGRESS     ; Lhs = logg ; Rhs = x  $
WALD        ; Fn1 = b_logpnc-0
                ; Fn2 = b_logpuc-0
                ; Fn3 = b_logppt-0 $
? Compute the regression. Then use the saved matrices.
REGRESS     ; Lhs = logg ; Rhs = x  $
MATRIX      ; b2 = b(4:6)
                ; v22 = Varb(4:6,4:6)
                ; list    ; W = b2'<v22>b2  $
REGRESS     ; Lhs = logg ; Rhs = x ; Cls: b(4) – b(5) = 0 $
? Compute a restricted least squares estimator using matrix algebra.
MATRIX      ; r = [0,0,0,1,0,0,0 / 0,0,0,0,1,0,0 / 0,0,0,0,0,1,0] ; q = [0/0/0] $
MATRIX      ; bu = <X'X>*X'y ; C = R*<X'X>*R' ; d = R*b - q
                ; bc = bu - <X'X>*R'*<C>*d $
? Test the hypothesis that two coefficients are equal using a t test.
? This could be built directly into the REGRESS command with ; Cls:b(b(-b(5)=0.
REGRESS     ; Lhs = logg ; Rhs = x $
CALC ; list     ; tstat = (b(4)-b(5)) / sqr(varb(4,4)+varb(5,5)-2*varb(4,5))
                ; tc = ttb (.975, degfrdm)
                ; pvalue = 2*(1-tds(tstat,degfrdm)) $
```

```
? Chow test of structural change.  For each subperiod and for the full period, obtain the residual
? sum of squares (without displaying the whole regression).
        PERIOD      ; 1953-2004 $
        CALC        ; ssep = Ess(x,logg) ; k = col(x) $
        PERIOD      ; 1953 – 1973 $
        CALC        ; sse1 = Ess(x,logg) ; n1 = n $
        PERIOD      ; 1974 – 2004 $
        CALC        ; sse2 = Ess(x,logg) ; n2 = n $
        CALC ; list ; f = ((ssep - (sse1+sse2))/k) / ((sse1+sse2)/(n1+n2-2*k))
                    ; Ftb(.95,k,(n1+n2-2*k)) $
? Structural change test using Wald approach requires separate regressions and some
? matrix algebra.
        PERIOD      ; 1953 – 1973 $
? Compute regressions quietly as I am not interested in seeing the results.
        REGRESS     ; Lhs = logg ; Rhs = x ; quietly $
        MATRIX      ; b1 = b ; v1 = varb $
        PERIOD      ; 1974 – 2004 $
        REGRESS     ; Lhs = logg ; Rhs = x ; quietly $
        MATRIX      ; b2 = b ; v2 = varb $
? Compute Wald statistic.  Then display critical value from chi squared table.
        MATRIX      ; db = b1 – b2 ; vdb = v1 + v2
                ; list  ; w = db'<vdb>db $
        CALC    ; list  ; cstar = ctb(.95,k) $
```

6.5 Using LIMDEP for Linear Regression Analysis

The following analyses show how to use *LIMDEP* for the standard applications in analysis of the linear model.

6.5.1 Obtain Descriptive Statistics

The command for obtaining descriptive statistics is

```
DSTAT       ; Rhs = the variables to be described $
```

There is a useful short hand; the 'wildcard' character '*' in this context means 'all variables.' So, we will use the simpler instruction,

```
DSTAT       ; Rhs = * $
```

The results are shown in the output window shown in Figure 6.8.

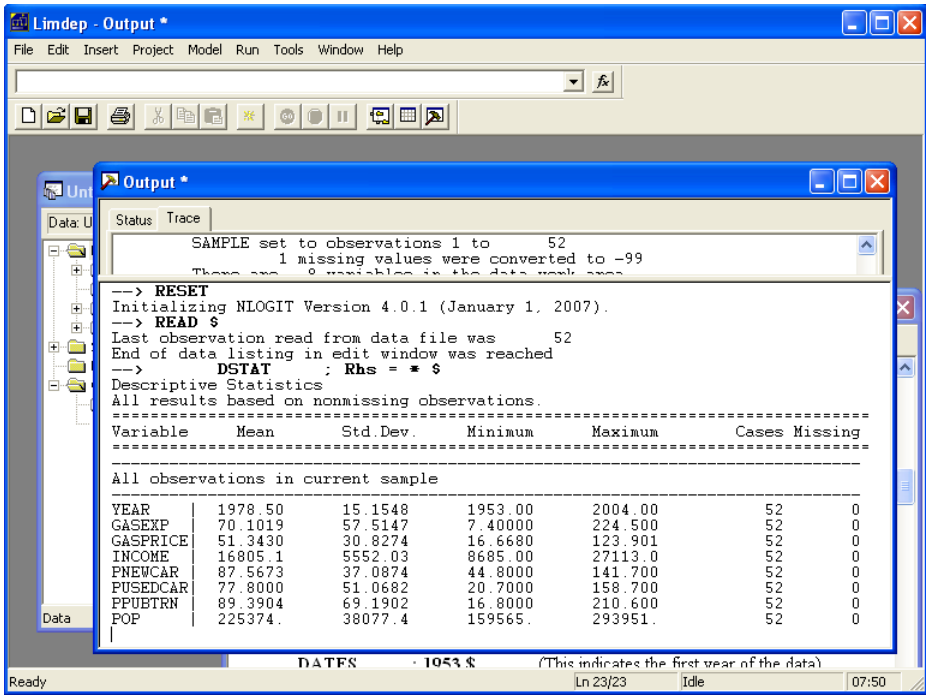


Figure 6.8 Descriptive Statistics in the Output Window

6.5.2 Transformed Variables

The command for computing transformed variables is **CREATE**. A single command can be used for all of the transformations listed,

```
CREATE    ; logg   = log (gasexp/(pop*gasprice))
          ; logpg  = log(gasprice)
          ; logi   = log(income)
          ; logpnc = log(pnewcar)
          ; logpuc = log(pusedcar)
          ; logppt = log(ppubtrn)
          ; t      = Year – 1953 $
```

After we place this command in the text editor and execute it, the project expands to include the new variables. The new project window in shown in Figure 6.9.

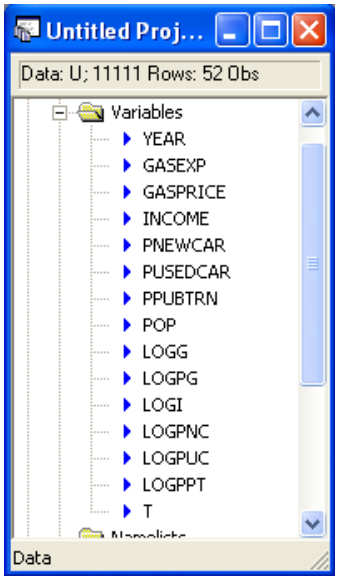


Figure 6.9 Project Window

6.5.3 Saving and Retrieving Your Project

Since you have read your data into the program and created some new variables, this is a good time to save your project. Select File:Save Project As… The mini-explorer that appears next allows you to save the file anywhere you like. The Project Files folder is a natural choice. See Figures 6.10 and 6.11 for our application.

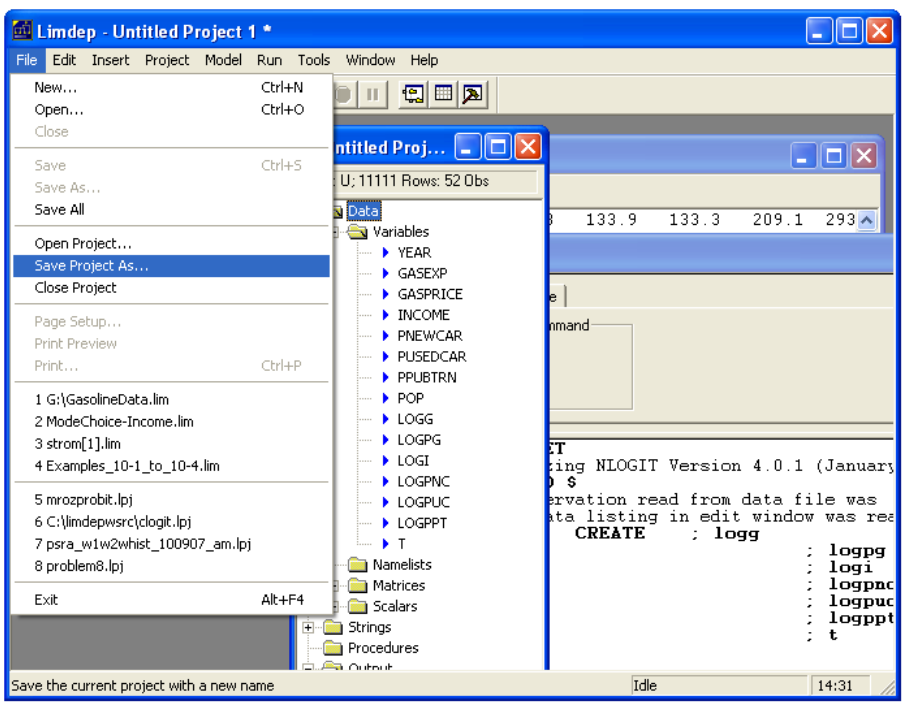


Figure 6.10 Saving the Current Project

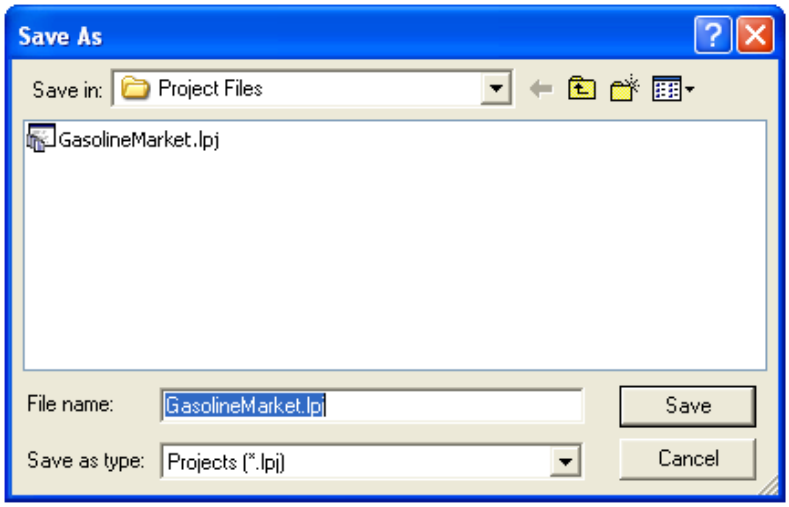


Figure 6.11 Mini-explorer for Saving the Project

When you restart *LIMDEP* later, you can use the File menu to retrieve your project file and resume your work where you left off before. The File menu is shown in Figure 6.12. Note that once we have saved our project file, the file name will reappear in the Recently Used part of the File menu.

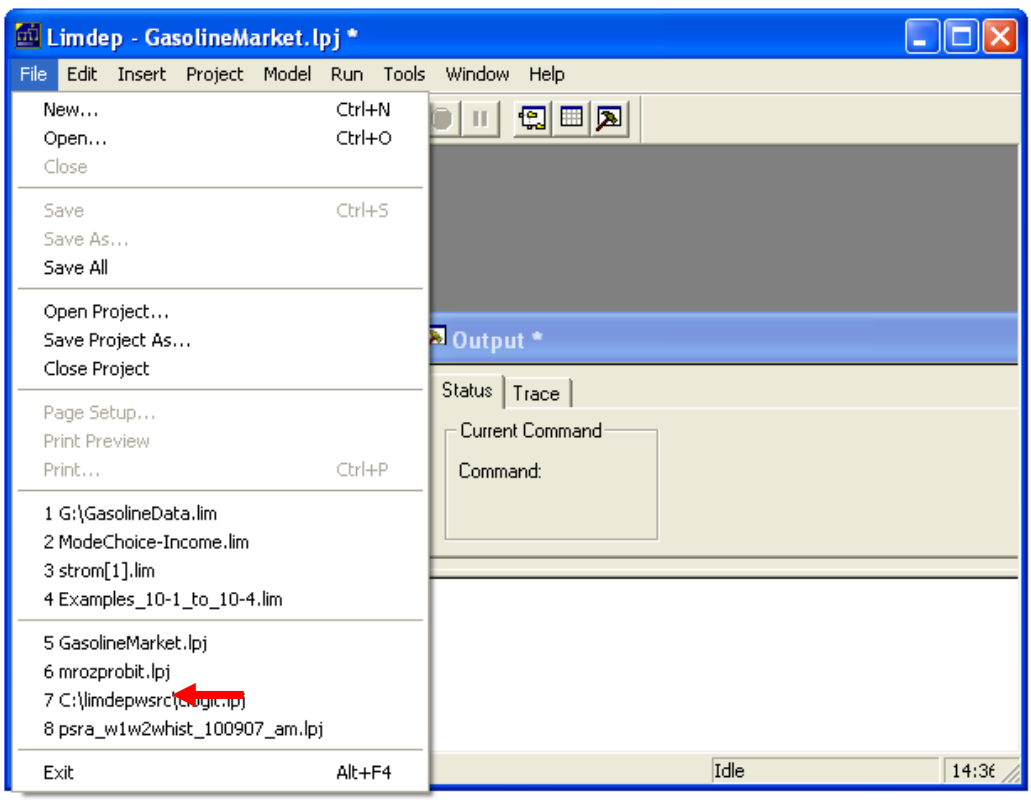


Figure 6.12 Reloading the Project

As with any software, it is a good idea to save your work this way periodically.

6.5.4 Time Series Plot of the Price Variables

The **PLOT** instruction is used to produce the different types of graphs you will obtain. Since these are time series data, we will first inform *LIMDEP* of that fact. The commands are

DATES ; 1953 $ (This indicates the first year of the data)
PERIOD ; 1953 – 2004 $ (This indicates which years of data to use)

Then, we generate our plot with the command

```
PLOT      ; Rhs = gasprice, pnewcar, pusedCar, ppubtrn
          ; Grid
          ; Title=Time Series Plot of Price Indices $
```

It is clear from Figure 6.13 that although the price series are correlated – that is in the nature of aggregate long period time series data – the correlations between the gasoline price and the public transport price seem unlikely to be severe enough to seriously impact the regression. However, the new and used car price indexes are quite highly correlated, and one might surmise that together in a regression, it would be difficult to resolve separate impacts of these two variables.

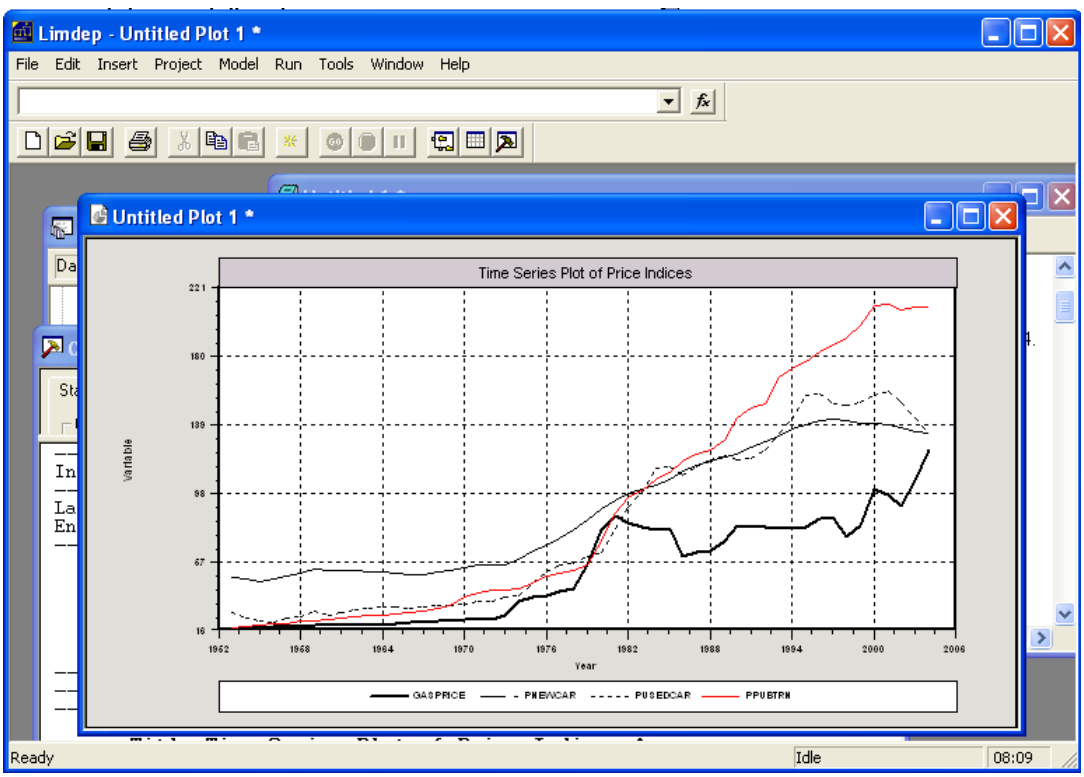


Figure 6.13 Time Series Plots

To pursue the issue a bit, we will compute a correlation matrix for the four price variables. As usual, there is more than one way to proceed. Here are two: First, the **DSTAT** command can be extended to request correlations by simply adding **; Output = 2** to the instruction. Thus, we might use

DSTAT ; Rhs = gasprice,pnewcar,pusedcar,ppubtrn ; Output = 2 $

This produces the results in Figure 6.14; the descriptive statistics now include the correlations among the price variables.

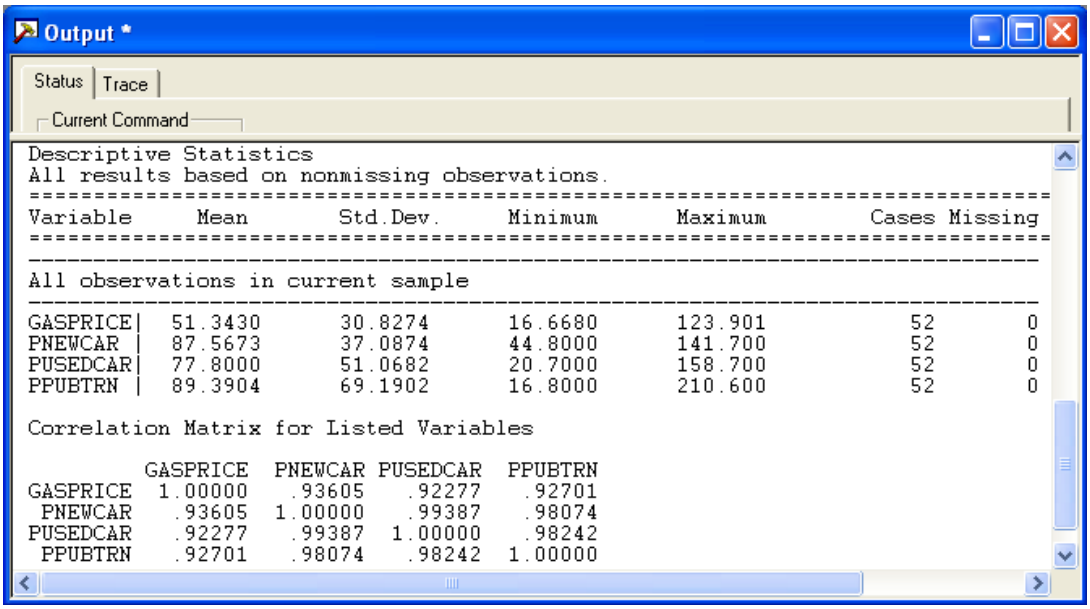


Figure 6.14 Descriptive Statistics with Correlations

It is possible to obtain these results with the dialog boxes in the command builder. We begin with Model:Data Description:Descriptive Statistics on the desktop menu shown in Figures 6.15 and 6.16.

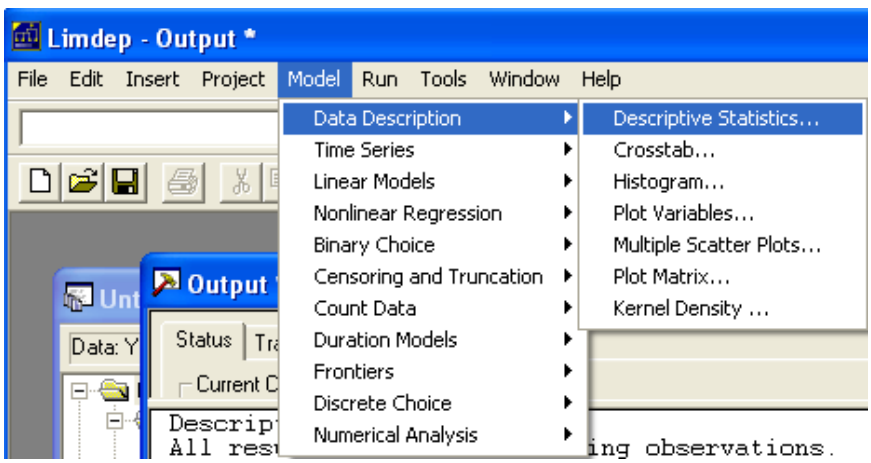


Figure 6.15 Model Menu for Descriptive Statistics

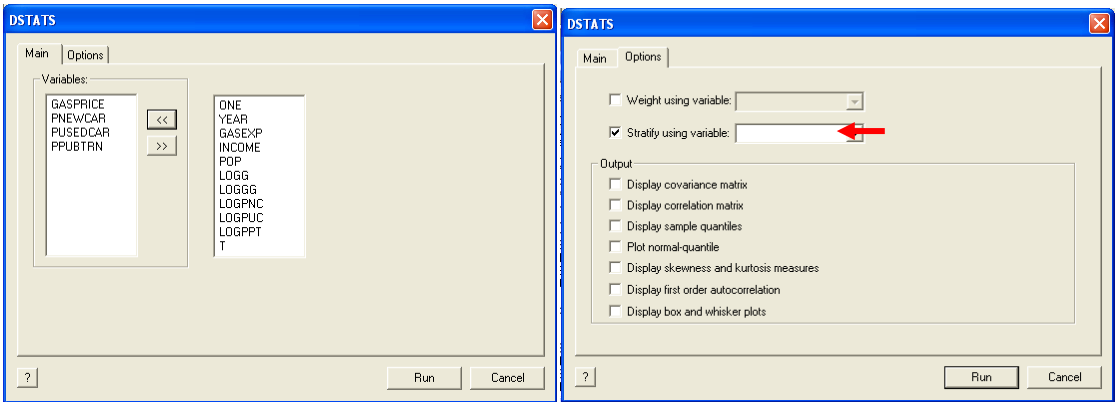


Figure 6.16 Command Builder for Descriptive Statistics

The variables are specified on the Main page, then the correlation matrix is requested as an option on the Options page. Clicking Run then produces the results that appear in Figure 6.14.

A second approach is to compute and display the correlation matrix. To do this, since we are computing a correlation matrix, we use the **MATRIX** command. There is a matrix function that computes correlations, which would appear as follows:

```
MATRIX      ; list ; Xcor (gasprice,pnewcar,pusedcar,ppubtrn) $
```

This produces the following results in the output window.

```
Correlation Matrix for Listed Variables
          GASPRICE  PNEWCAR PUSEDCAR  PPUBTRN
GASPRICE  1.00000   .93605   .92277   .92701
 PNEWCAR   .93605  1.00000   .99387   .98074
PUSEDCAR   .92277   .99387  1.00000   .98242
 PPUBTRN   .92701   .98074   .98242  1.00000
```

which are, of course the same as those with the descriptive statistics.

There is another convenient feature of **MATRIX** that we should note at this point. The **NAMELIST** command below associates the name ‘prices’ with the four price variables:

```
NAMELIST    ; prices = gasprice,pnewcar,pusedcar,ppubtrn $
```

Now, in any setting where we wish to use these four variables, we can use the name prices instead. This defines a data matrix with these four columns. Thus, we can now shorten the MATRIX command to

```
MATRIX      ; list ; Xcor (prices) $
```

Finally, note that each **MATRIX** command begins with **;list**. This requests that the result of the matrix computation be displayed on the screen in the output window. Why is this needed? **MATRIX** is part of the programming language. You might be writing sets of commands that do matrix computations that are intermediate results to be used later that you are not necessarily interested in seeing. If you do not request *LIMDEP* to list the results, the program assumes this is the case. Several examples below will illustrate this.

6.5.5 Simple Regression

The REGRESS command is used to compute a linear regression. (As usual, you can also use the Model menu and command builders, however, we will generally use the commands.) The essential instruction is

REGRESS ; Lhs = one,logpg $

The regression results are displayed in the output window shown in Figure 6.17.

NOTE: The right hand side of the **REGRESS** command contains two 'variables,' *one* and the variable that we wish to appear in the model. The '*one*' is the constant term in the model. If you wish for your model to include a constant term – and this should be in the vast majority of cases – you request it by including one on the right hand side. *LIMDEP* does not automatically include a constant term in the model. In fact, most programs work this way, though there are a few that automatically put a constant term in the model. Then, it becomes inconvenient to fit a model without one. In general, *LIMDEP* requires you to specify the model the way you want it; it does not make assumptions for you.

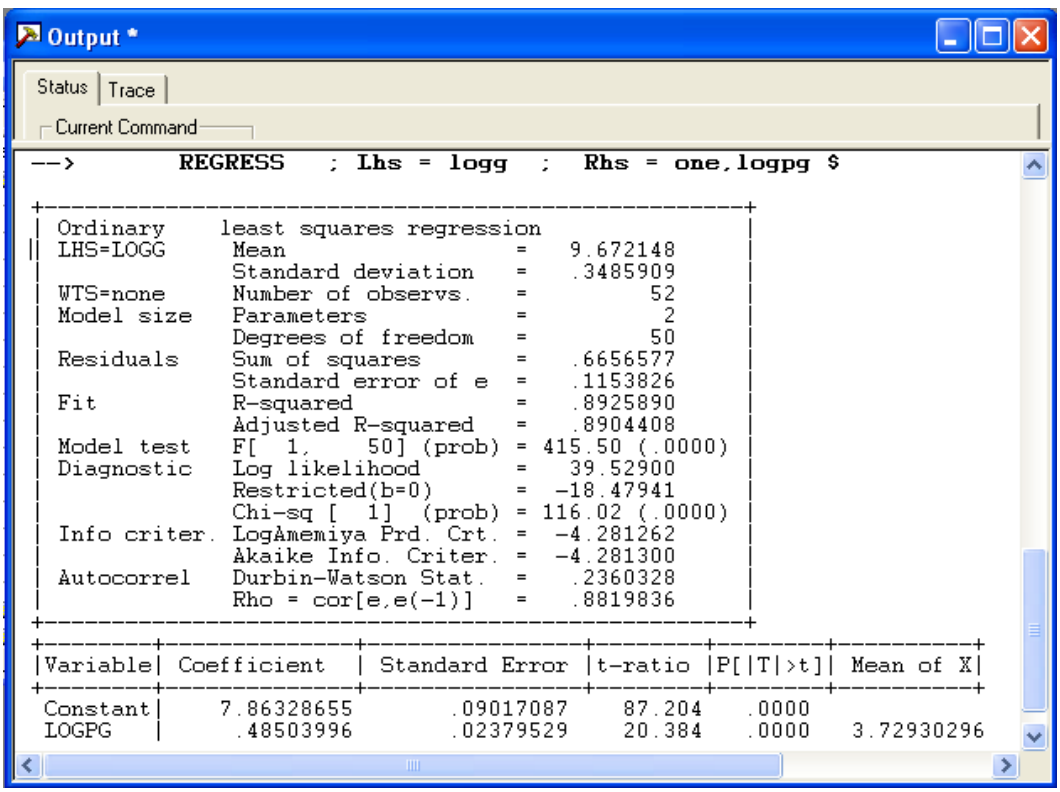


Figure 6.17 Regression Results

The regression output shows *LIMDEP*'s standard format. The set of diagnostics, including R^2, s^2, sum of squares, F statistic for the regression, and so on are shown above the regression coefficients. The coefficients are reported with standard errors, *t* ratios, '*p* values' and the means of the associated independent variables.

For the specific application, we note, the estimated price elasticity is, indeed, positive, which is unexpected for economic data. Two explanations seem natural. First, assuming we are estimating a demand equation, then a demand equation should include income as well as price. In this particular market, in fact, it is well known that demand is strikingly inelastic with respect to price, and that income is the primary driver of consumption. The estimated equation is missing a crucial variable. The theoretical result for a demand equation from which income has been omitted would be as follows:

Model: $logG = \alpha + \beta_{Price}\, logPrice + \beta_{Income}\, logIncome + \varepsilon$

$$\text{plim } b_{Price} = \beta_{Price} + \frac{\text{Cov}(logPrice, logIncome)}{\text{Var}(logPrice)}\beta_{Income}$$

where b_{Price} is the simple least squares slope in the regression of *logG* on a constant and *logPg*. It is clear from the data that the two terms in the fraction are positive. If the income elasticity were positive as well, then the result shows that the estimator in this short regression is biased upward toward and possibly even past zero. We could construct some evidence on what might be expected here. Suppose the income elasticity were +1.0 and the price elasticity were -0.1. The terms in the fraction can be obtained with the command

```
CALC  ; list    ; spi = Cov(logpg,logi)
                ; spp  = Var(logpg)
                ; plim = -0.1 + spi/spp * 1.0 $
```

The result is shown here. If the theory is right, then even if the price elasticity really is negative, with these time series data and the positive income elasticity, we should observe a positive coefficient on *logPg* in the simple regression (which we did).

```
+-------------------------------------+
| Listed Calculator Results           |
+-------------------------------------+
 SPI     =       .223618
 SPP     =       .461029
 PLIM    =       .385040
Calculator: Computed   3 scalar results
```

We proceed to compute the more appropriate regression which contains both price and income (and which will either confirm or refute our theory above). We do note the second explanation for the finding, however. We have presumed so far that we are actually fitting a demand curve, and that we can treat the price as exogenous. If the price were determined on a world market, this might be reasonable, but that is less so in the U.S. which does manipulate the price of gasoline. We leave for others to resolve this issue, and continue with the multiple regression.

The more complete model is estimated with the command

```
REGRESS      ; Lhs = logG ; Rhs = one,logpg,logi $
```

The results shown below are more in line with expectations, and conform fairly closely to the hypothetical exercise done earlier. The data from this period do suggest that the income elasticity is close to one and the price elasticity is indeed negative and around -0.17. The fit of the model is extremely good as well.

```
+----------------------------------------------------+
| Ordinary    least squares regression               |
| LHS=LOGG     Mean                 =  -12.24504     |
|              Standard deviation   =   .2388115     |
| WTS=none     Number of observs.   =         52     |
| Model size   Parameters           =          3     |
|              Degrees of freedom   =         49     |
| Residuals    Sum of squares       =   .1849006     |
|              Standard error of e  =   .6142867E-01 |
| Fit          R-squared            =   .9364292     |
|              Adjusted R-squared   =   .9338345     |
| Model test   F[  2,    49] (prob) = 360.90 (.0000) |
| Diagnostic   Log likelihood       =   72.83389     |
|              Restricted(b=0)      =   1.188266     |
|              Chi-sq [  2]  (prob) = 143.29 (.0000) |
| Autocorrel   Durbin-Watson Stat.  =   .1168578     |
|              Rho = cor[e,e(-1)]   =   .9415711     |
+----------------------------------------------------+
+--------+--------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |t-ratio |P[|T|>t]| Mean of X|
+--------+--------------+----------------+--------+--------+----------+
 Constant|   -20.9557732       .59398134   -35.280   .0000
 LOGPG   |    -.16948546       .03865426    -4.385   .0001   3.72930296
 LOGI    |     .96594886       .07529145    12.829   .0000   9.67214751
```

As a final check on the model, we have plotted the residuals by adding **; PlotResiduals** to the **REGRESS** command.

REGRESS ; Lhs = logG ; Rhs = one,logpg,logi ; Plot residuals $

The results are striking;. The residuals are far from a random sequence. The long sequences of negative, then positive, then negative residuals suggests that something is clearly missing from the equation.

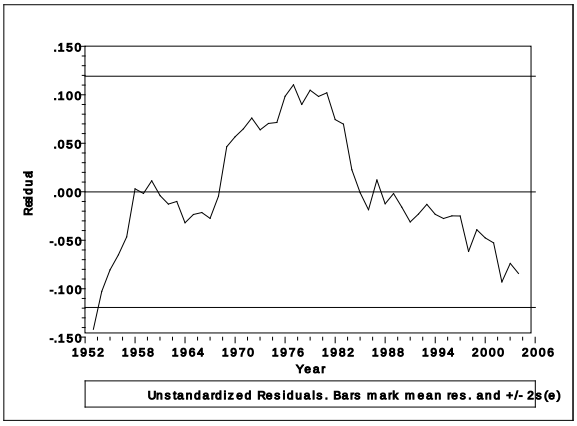


Figure 6.18 Residual Plot

6.5.6 Multiple Regression

For the full model, we use the following setup, which combines some of our earlier commands.

```
NAMELIST    ; x1 = logpg,logI
            ; x2 = logpnc,logpuc,logppt
            ; x =one, x1,x2,t $
```

Notice the third namelist is constructed by combining the first two and adding a variable to the list. The regression is then computed using

```
REGRESS     ; Lhs = logg ; Rhs = x ; Plot residuals $
```

The full results are as follows:

```
+----------------------------------------------------+
| Ordinary    least squares regression               |
| LHS=LOGG     Mean                 =  -12.24504     |
|              Standard deviation   =   .2388115     |
| WTS=none     Number of observs.   =         52     |
| Model size   Parameters           =          7     |
|              Degrees of freedom   =         45     |
| Residuals    Sum of squares       =   .1014368     |
|              Standard error of e  =   .4747790E-01 |
| Fit          R-squared            =   .9651249     |
|              Adjusted R-squared   =   .9604749     |
| Model test   F[  6,    45] (prob) = 207.55 (.0000) |
| Diagnostic   Log likelihood       =   88.44384     |
|              Restricted(b=0)      =   1.188266     |
|              Chi-sq [  6]  (prob) = 174.51 (.0000) |
| Autocorrel   Durbin-Watson Stat.  =   .4470769     |
|              Rho = cor[e,e(-1)]   =   .7764615     |
+----------------------------------------------------+
+--------+-------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |t-ratio |P[|T|>t]| Mean of X|
+--------+-------------+----------------+--------+--------+----------+
 Constant|   -26.9680492      2.09550408   -12.869   .0000
 LOGPG   |    -.05373342       .04251099    -1.264   .2127   3.72930296
 LOGI    |    1.64909204       .20265477     8.137   .0000   9.67214751
 LOGPNC  |    -.03199098       .20574296     -.155   .8771   4.38036654
 LOGPUC  |    -.07393002       .10548982     -.701   .4870   4.10544881
 LOGPPT  |    -.06153395       .12343734     -.499   .6206   4.14194132
 T       |    -.01287615       .00525340    -2.451   .0182   25.5000000
```

The coefficient on *logPg* has now fallen to only -0.05. The implication is that after accounting for the uses of gasoline and the price of a major competitor (public transport) and income, the price elasticity of demand for gasoline really does seem to be close to zero. The residuals are still far from random, but are somewhat more in that direction than the earlier ones. This suggests, as surmised earlier, that the specification of the model is in need of improvement.

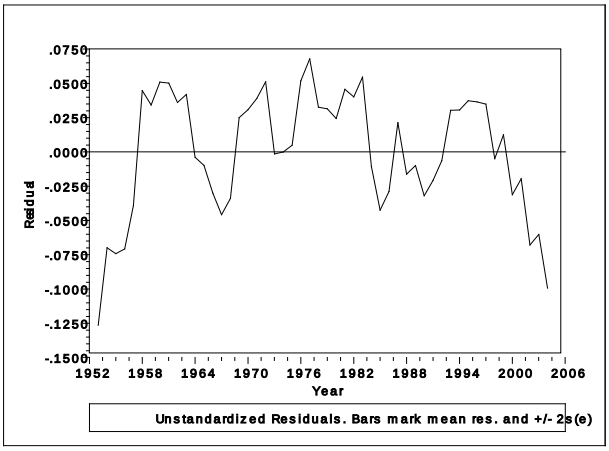


Figure 6.19 Residual Plot from Expanded Model

The least squares coefficients, standard errors, *s* and R^2 are all reported with the standard results. It is also convenient to obtain these results using least squares algebra, as shown below.

```
MATRIX    ; bols = <X'X> * X  logg $
MATRIX    ; e  = logg - X * bols $
CALC  ; list ; se  =  sqr((e'e) / (n-col(X)))
             ; r2 = 1  - e'e/((n-1)*var(logg))$
MATRIX    ; v = s2 * <X'X>
             ; Stat (bols,v,x) $
```

(Note that in this set of commands, we are avoiding using the reserved names *s* and *rsqrd*.)

```
+-----------------------------------+
| Listed Calculator Results         |
+-----------------------------------+
 SE      =       .047478
 R2      =       .965125
+---------------------------------------------------+
|Number of observations in current sample =      52 |
|Number of parameters computed here       =       7 |
|Number of degrees of freedom             =      45 |
+---------------------------------------------------+
+--------+-------------+----------------+--------+--------+
|Variable| Coefficient  | Standard Error |b/St.Er.|P[|Z|>z]|
+--------+-------------+----------------+--------+--------+
 Constant|   -26.9680492      2.09550408   -12.869   .0000
 LOGPG   |    -.05373342       .04251099    -1.264   .2062
 LOGI    |    1.64909204       .20265477     8.137   .0000
 LOGPNC  |    -.03199098       .20574296     -.155   .8764
 LOGPUC  |    -.07393002       .10548982     -.701   .4834
 LOGPPT  |    -.06153395       .12343734     -.499   .6181
 T       |    -.01287615       .00525340    -2.451   .0142
```

To test the hypothesis that all the coefficients save for the constant term are zero, we need

$$F[K\text{-}1,n\text{-}K] = [R^2/K] / [(1\text{-}R^2)/(n\text{-}K)]$$

The value is reported in the regression results above (207.55). But, it is easy enough to compute it directly, using

```
CALC      ; list ; f = (r2/(col(X)-1)) / ((1-r2)/(n-col(X))) $
```

```
+-----------------------------------+
| Listed Calculator Results         |
+-----------------------------------+
 F       =    207.553435
```

In fact, this can be made a bit easier, because the ingredients we need were automatically computed and retained by the regression command. After the **REGRESS** command is executed, the new project window is

The new scalars that appear in the project window are

ssqrd	$= s^2$
rsqrd	$= R^2$
s	$= s$
sumsqdev	$=$ **e′e**
degfrdm	$= n - K$
kreg	$= K$
nreg	$= n$

for the most recent regression computed. Thus, after the regression command, we could have used

```
CALC  ; list  ; se  =  sqr(sumsqdev /degfrdm)
              ; r2 = 1  - e'e/((n-1)*var(logg)) $
```

Of course, se and r2 were superfluous, since s and rsqrd are the same values. For computing the F statistic, we could have used

```
CALC      ; 5 = rsqrd/(k-1)/((1-rsqrd)/degfrdm) $
```

Finally, in order to obtain the critical value for the F test, we use the internal table. For the F statistic and t statistics, we use

```
CALC  ; list ; fc = Ftb(.95,(kreg-1),degfrdm)
              ; tc  =  ttb(.95, (n-col(x)))$
```

which produces

```
+-----------------------------------+
| Listed Calculator Results         |
+-----------------------------------+
 FC      =      2.308273
 TC      =      2.014103
```

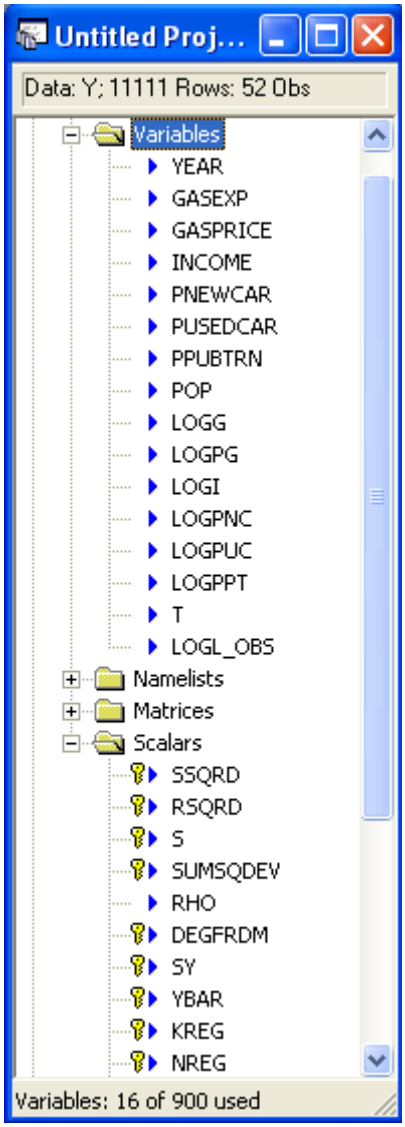


Figure 6.20 Project Window

6.5.7 Hypothesis Tests

There are usually several ways to carry out a computation. Here we will test a hypothesis using several tools. The model is

$$logG = \beta_1 + \beta_2 logPg + \beta_3 logI + \beta_4 logPnc + \beta_5 logPuc + \beta_6 logPpt + \beta_7 t + \varepsilon$$

The first hypothesis to be tested is that the three cross price elasticities are all zero. That is,

$$H_0: \beta_4 = \beta_5 = \beta_6 = 0.$$

A direct approach is to build the linear hypothesis directly into the model command. There is a particular specification for this, as shown in the command below:

```
NAMELIST   ; x1 = logpg,logi  ; x2 = logpnc,logpuc,logppt ; x =one, x1,x2,t $
REGRESS    ; Lhs = logg ; Rhs = x
           ; Cls: b(4) = 0, b(5) = 0, b(6) = 0 $
```

This produces the following results: The results of the unrestricted regression are shown first, followed by

```
+----------------------------------------------------+
| Linearly restricted regression                     |
| Ordinary    least squares regression               |
| LHS=LOGG     Mean                 =  -12.24504     |
|              Standard deviation   =   .2388115     |
| WTS=none     Number of observs.   =         52     |
| Model size   Parameters           =          4     |
|              Degrees of freedom   =         48     |
| Residuals    Sum of squares       =   .1193176     |
|              Standard error of e  =   .4985762E-01 |
| Fit          R-squared            =   .9589773     |
|              Adjusted R-squared   =   .9564134     |
| Model test   F[  3,    48] (prob) = 374.03 (.0000) |
| Diagnostic   Log likelihood       =   84.22267     |
|              Restricted(b=0)      =   1.188266     |
|              Chi-sq [  3]  (prob) = 166.07 (.0000) |
| Autocorrel   Durbin-Watson Stat.  =   .4220089     |
|              Rho = cor[e,e(-1)]   =   .7889955     |
| Restrictns.  F[  3,    45] (prob) =   2.64 (.0606) |
| Not using OLS or no constant. Rsqd & F may be < 0. |
| Note, with restrictions imposed,  Rsqd may be < 0. |
+----------------------------------------------------+
+--------+-------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |t-ratio |P[|T|>t]| Mean of X|
+--------+-------------+----------------+--------+--------+----------+
 Constant|   -28.0817710      1.46871226   -19.120   .0000
 LOGPG   |    -.13016036       .03229378    -4.031   .0002   3.72930296
 LOGI    |    1.73922391       .16247609    10.704   .0000   9.67214751
 LOGPNC  |    .249800D-15   ......(Fixed Parameter).......
 LOGPUC  |   -.555112D-16   ......(Fixed Parameter).......
 LOGPPT  |   -.117961D-15   ......(Fixed Parameter).......
 T       |    -.01960344       .00381652    -5.136   .0000   25.5000000
```

(The coefficients on the three prices in the restricted regression are not exactly zero – the values are pure rounding error.) The *F* statistic is on the boundary, not quite statistically significant.

Although the restrictions can be built into the **REGRESS** command, there will be cases in which one is interested in applying the theoretical results directly. The template formula for computing the *F* statistic is

$$F = \frac{(R^2_{\text{Unrestricted}} - R^2_{\text{Restricted}}) / J}{(1 - R^2_{\text{Unrestricted}}) / (n - K)}$$

Where *J* is the number of restrictions and *K* is the number of parameters in the full regression with no restrictions. In order to compute the statistic this way, we need the R^2s from the two regressions. For this application, the restricted regression omits the three additional price variables. A way we can proceed is

```
NAMELIST    ; x1 = logpg,logI ; x2 = logpnc,logpuc,logppt  ; x =one, x1,x2,t
            ; xu = x  ;  xr = one,x1,t $
REGRESS     ; Lhs = logg ; Rhs = xu $
CALC        ; rsqu = rsqrd $
REGRESS     ; Lhs = logg ; Rhs = xr $
CALC        ; rsqr = rsqrd
      ; List ; f = ((rsqu – rsqr)/Col(x2))/((1-rsqu)/(n-Col(x))) $
```

Finally, since we aren't actually interested in seeing the regressions at this point, there is a more direct way to proceed. The calculator function Rsq(matrix,variable) computes the R^2 in the regression of the variable on the variables in the matrix, which is exactly what we need. Thus,

```
CALC        ; rsqu = Rsq(xu,logg) ; rsqr = Rsq(xr,logg)
      ; List ; f = ((rsqu – rsqr)/Col(x2))/((1-rsqu)/(n-Col(x))) $
```

An alternative way to compute the test statistic is to use the large sample version, which is a chi squared statistic, the Wald statistic. Formally, the Wald statistic is

$$W = \mathbf{b}_2' (\mathbf{W}_{22})^{-1} \mathbf{b}_2$$

Where $\mathbf{b}_2$ is the subset of the least squares coefficient vector, the three price coefficients, and $\mathbf{W}_{22}$ is the 3×3 submatrix of the covariance matrix of the coefficient estimator. This is estimated with $s^2(\mathbf{X'X})^{-1}$. For the linear regression model, the *W* statistic turns out to be just *J* times the *F* statistic, where J is the number of restrictions (three). We can compute this by using the built in program written just for this purpose, as follows:

```
REGRESS     ; Lhs = logg ; Rhs = x  $
WALD        ; Fn1 = b_logpnc-0
            ; Fn2 = b_logpuc-0
            ; Fn3 = b_logppt-0 $
```

The **WALD** procedure is used for two purposes, first, to compute estimates of standard errors for functions of estimated parameters and, second, to compute Wald statistics. The result for our model is

```
+---------------------------------------------+
| WALD procedure. Estimates and standard errors |
| for nonlinear functions and joint test of     |
| nonlinear restrictions.                       |
| Wald Statistic             =      7.93237     |
| Prob. from Chi-squared[ 3] =       .04743     |
+---------------------------------------------+
+--------+-------------+---------------+--------+--------+
|Variable| Coefficient  | Standard Error |b/St.Er.|P[|Z|>z]|
+--------+-------------+---------------+--------+--------+
 Fncn(1) |    -.03199098       .20574296     -.155   .8764
 Fncn(2) |    -.07393002       .10548982     -.701   .4834
 Fncn(3) |    -.06153395       .12343734     -.499   .6181
```

The chi squared statistic is given in the box of results above the estimates of the functions. The statistic given is the statistic for the joint test of the hypothesis that the three functions equal zero.

To illustrate the computations, we consider how to obtain the Wald statistic 'the hard way,' that is, by using matrix algebra. This would be

```
REGRESS     ; Lhs = logg ; Rhs = x  $
MATRIX      ; b2 = b(4:6)  ; v22 = Varb(4:6,4:6)
            ; list     ; w = b2'<v22>b2  $
```

```
Matrix W has  1 rows and  1 columns.
               1
        +--------------
       1|    7.93237
```

It is also instructive to see how to compute the restricted least squares estimator using the textbook formulas. The result for computing the least squares estimator **b*** subject to the restrictions **Rb –q** = **0** is

$$\mathbf{b^*} = \mathbf{b} - (\mathbf{X'X})^{-1}\mathbf{R'}[\mathbf{R}(\mathbf{X'X})^{-1}\mathbf{R'}]^{-1}(\mathbf{Rb} - \mathbf{q})$$

where b is the unconstrained estimator. To compute this using matrix algebra, we use

```
MATRIX      ; r = [0,0,0,1,0,0,0 / 0,0,0,0,1,0,0 / 0,0,0,0,0,0,1,0] ; q = [0/0/0] $
MATRIX      ; bu = <X'X>*X'y ; c = r*<X'X>*r' ; d = r*b - q
            ; bc = bu - <X'X>*r'*<c>*d $
```

The second hypothesis to be tested is that the elasticities with respect to new and used cars are equal to each other, which is

H_0: $\beta_4 = \beta_5$.

To carry out this test using the *F* statistic, we can use any of the devices we used earlier. The most straightforward would be to place the restriction in the **REGRESS** command;

REGRESS ; Lhs = logg ; Rhs = x ; Cls: b(4) – b(5) = 0 $

The regression results with this restriction imposed are shown below. In order to carry out the test using a t statistic, we will require

$$t = \frac{b_4 - b_5}{\sqrt{v_{44} + v_{55} - 2v_{45}}}$$

Where v_{kl} is the estimated covariance of b_k and b_l. After computing the unrestricted regression, we can use CALC to obtain this result:

```
REGRESS      ; Lhs = logg ; Rhs = x $
CALC; List   ; tstat = (b(4)-b(5)) / sqr(varb(4,4)+varb(5,5)-2*varb(4,5))
             ; tc = ttb (.975, degfrdm)
             ; pvalue = 2*(1-tds(tstat,degfrdm)) $
```

```
+-----------------------------------------------------+
| Linearly restricted regression                      |
| LHS=LOGG     Mean                 =  -12.24504      |
|              Standard deviation   =   .2388115      |
| WTS=none     Number of observs.   =         52      |
| Model size   Parameters           =          6      |
|              Degrees of freedom   =         46      |
| Residuals    Sum of squares       =   .1014853      |
|              Standard error of e  =   .4697022E-01 |
| Fit          R-squared            =   .9651083      |
|              Adjusted R-squared   =   .9613157      |
| Model test   F[  5,    46] (prob) = 254.47 (.0000) |
| Autocorrel   Durbin-Watson Stat.  =   .4411613      |
|              Rho = cor[e,e(-1)]   =   .7794194      |
| Restrictns.  F[  1,    45] (prob) =    .02 (.8841) | <-
| Not using OLS or no constant. Rsqd & F may be < 0.  |
| Note, with restrictions imposed,  Rsqd may be < 0.  |
+-----------------------------------------------------+
+--------+--------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |t-ratio |P[|T|>t]| Mean of X|
+--------+--------------+----------------+--------+--------+----------+
 Constant|   -26.7523913      1.47691600   -18.114   .0000
 LOGPG   |    -.05231601       .04095506    -1.277   .2080   3.72930296
 LOGI    |    1.63099549       .15903625    10.255   .0000   9.67214751
 LOGPNC  |    -.06096105       .05689811    -1.071   .2897   4.38036654
 LOGPUC  |    -.06096105       .05689811    -1.071   .2897   4.10544881
 LOGPPT  |    -.05636622       .11703582     -.482   .6324   4.14194132
 T       |    -.01262751       .00491914    -2.567   .0137   25.5000000

+-----------------------------------+
| Listed Calculator Results         |
+-----------------------------------+
 TSTAT   =       .146654
 TC      =      2.014103
 PVALUE  =       .884061
```

6.5.8 Test of Structural Break

The 'Chow test' is used to determine if the same model should apply to two (or more) subsets of the data. Formally, the test is carried out with the F statistic,

$$F[K, n_1 + n_2 - 2K] = \frac{(SSE_{Pooled} - (SSE_1 + SSE_2))/K}{(SSE_1 + SSE_2)/(n_1 + n_2 - 2K)}$$

Where SSE_j is the sum of squared residuals from the indicated regression and K is the number of coefficients (variables plus the constant) in the model. Superficially, this requires us to compute the three regressions, then compute the statistic. We might not be interested in seeing the actual regression results for the subperiods, so we go directly to the sum of squares function using the calculator. We proceed as follows:

```
PERIOD          ; 1953-2004 $
CALC            ; ssep = Ess(x,logg) ; k = col(x) $
PERIOD          ; 1953 – 1973 $
CALC            ; sse1 = Ess(x,logg) ; n1 = n $
PERIOD          ; 1974 – 2004 $
CALC            ; sse2 = Ess(X,logg) ; n2 = n $
CALC ; list     ; f = ((ssep - (sse1+sse2))/k) / ((sse1+sse2)/(n1+n2-2*k))
                ; fc = Ftb(.95,k,(n1+n2-2*k)) $
```

The results shown are the sample *F* and the critical value from the *F* table. The null hypothesis of equal coefficient vectors is clearly rejected.

```
+-----------------------------------+
| Listed Calculator Results         |
+-----------------------------------+
 F       =     83.664571
 FC      =      2.262304
```

The second approach suggested is to use a Wald test to test the hypothesis of equal coefficients. For the structural change test, the statistic would be

$$W = (\mathbf{b}_1 - \mathbf{b}_2)' [\mathbf{V}_1 + \mathbf{V}_2]^{-1} (\mathbf{b}_1 - \mathbf{b}_2).$$

The difference in the two approaches is that the Wald statistic does not assume that the disturbance variances in the two regressions are the same. We compute the test statistic using matrix algebra then obtain the critical value from the chi squared table.

```
PERIOD          ; 1953 – 1973 $
REGRESS         ; Lhs = logg ; Rhs = x ; quietly $
MATRIX          ; b1 = b ; v1 = varb $
PERIOD          ; 1974 – 2004 $
REGRESS         ; Lhs = logg ; Rhs = x ; quietly $
MATRIX          ; b2 = b ; v2 = varb $
MATRIX          ; db = b1 – b2 ; vdb = v1 + v2 ; list ; W = db'<vdb>db $
CALC  ; list    ; cstar = ctb(.95,k) $
```

The results of the Wald test are the same as the *F* test. The hypothesis is decisively rejected.

```
Matrix W        has  1 rows and  1 columns.
               1
        +--------------
       1|  612.92811
+-----------------------------------+
| Listed Calculator Results         |
+-----------------------------------+
 Result  =     14.067140
```

Chapter 7: Essentials of Data Management

7.1 Introduction

The tutorial in Chapter 6 showed how to read a simple data file, how to obtain transformed variables (logs), and how to change the sample by setting the period for time series data. There are many different options for these operations in *LIMDEP*. This chapter will describe a few of the additional features available in the program for managing your data. The sections to follow are:

7.2: Reading and Entering Data
7.3 Computing Transformed Variables
7.4: Lists of Variables
7.5: The Current Sample of Observations
7.6: Missing Data

7.2 Reading and Entering Data

Most of the analyses of externally created data that you do with *LIMDEP* will involve data sets that are entered via disk files. (The alternative is internally created data that you produce using *LIMDEP*'s random number generators.) The **READ** command is provided for this purpose.

7.2.1 The Data Area

Your data are stored in an area of memory that we will refer to as the *data array*. The data array in the student version of *LIMDEP* or *NLOGIT* contains 1,000 rows and 99 columns. These values are preset and cannot be changed.

7.2.2 The Data Editor

For initial entry of data, *LIMDEP*'s 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 entry Project:Data Editor.

The data editor is shown in Figure 7.1. (The chevrons next to the observation numbers indicate that these observations will be in the current sample. See Section 7.5.)

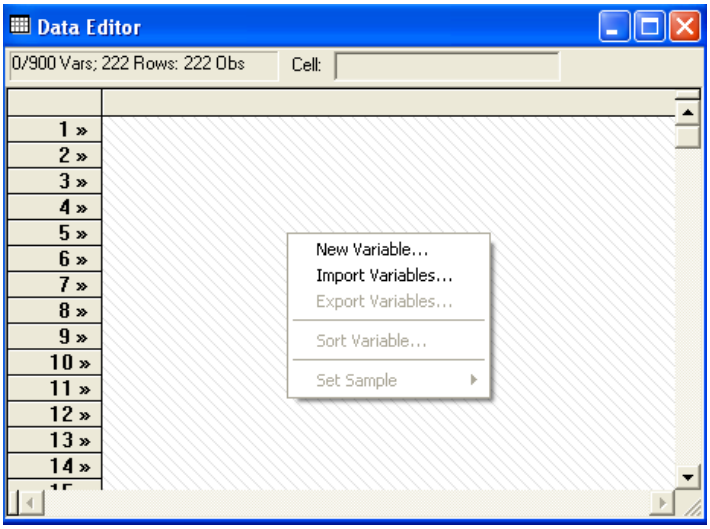


Figure 7.1 Data Editor

The data editor appears with an empty editing area when no variables exist. The functions of the data editor are shown in the smaller menu, which you obtain by pressing the *right* mouse button, displayed in Figure 7.1. The functions are:

New Variable

Click New Variable to open a dialog box that will allow you to create new variables with transformations as shown in Figure 7.2. (This is analogous to the **CREATE** command described in Section 7.3.) If you wish simply to type in a column of data – a variable – in the New Variable dialog box, just enter the name at the top of the box and click OK. This will create a new variable with a single observation equal to zero, which you will replace, and all remaining observations missing. You will then enter the data editor where you can type in the values in the familiar fashion.

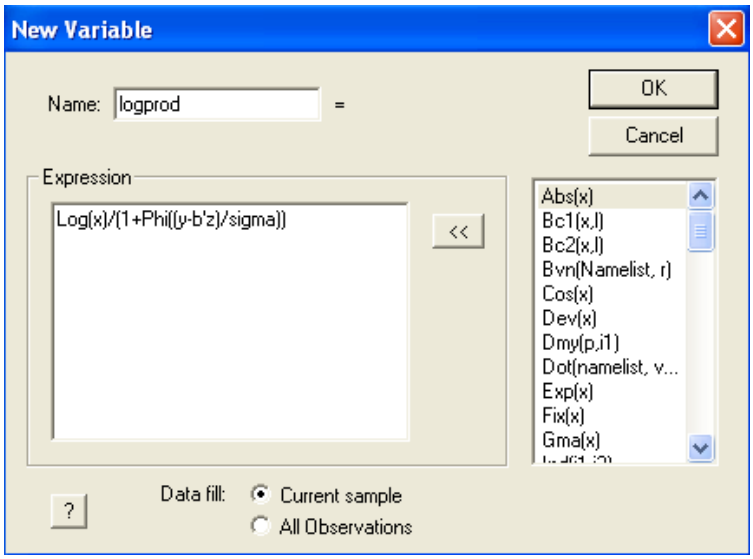


Figure 7.2 New Variable Dialog Box

The New Variable dialog box also allows you to transform existing variables after they have been entered. For example, in Figure 7.2, the dialog is being used to create a variable named *logprod* using two existing variables named *y* and *x* and the Log and Phi (normal CDF) functions.

After you create and enter new variables, the project window is updated, and the data, themselves, are placed in the data area. Figure 7.3 illustrates.

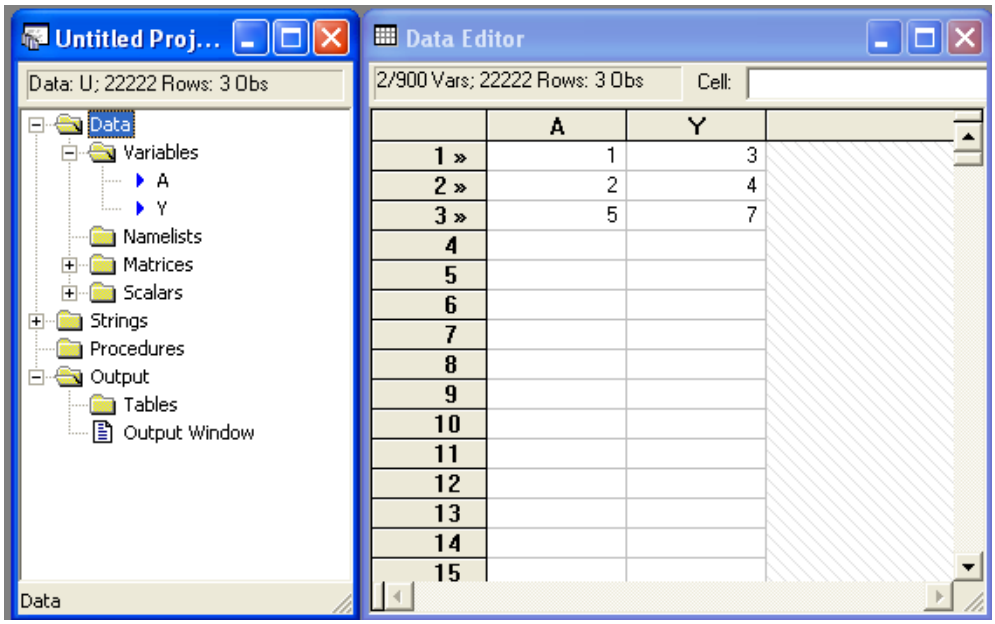


Figure 7.3 Data Editor and Project Window

Import Variables

Click Import Variables to open a dialog box that allows you to import a spreadsheet or other data file. This is analogous to the READ command described below.

Set Sample

Click Set Sample to obtain a menu of options for setting the current sample. This uses the **REJECT**, **INCLUDE** and **DATES** features discussed in Section 7.5 to set the current sample. (This option is not available until data have been read into the data area.)

HINT: The data editor does not automatically reset the current sample. After you enter a data set with it, the current sample is unchanged.

Data may be edited in the data editor in the usual fashion for spreadsheets. Note, however, that this means only entering new values or replacing old ones. We have not replicated the full transformation functionality in, e.g., *Excel*. Data may be transformed and manipulated using other features in *LIMDEP*. The New Variable... option, however, does provide all the features you might need to transform variables. Exit the editor simply by closing the window.

7.2.3 Reading Data Files Into *LIMDEP*

Data may be entered into *LIMDEP* from many different kinds of files. Formats include simple ASCII files, spreadsheet files written by other programs such as *Excel*, and binary as well as other types of files written by other statistical programs. The usual way of entering data in *LIMDEP* if you are not typing them in the data editor is in the form of ASCII text. An ASCII file is a text file that you can display in a word processing program. This section will describe how to read ASCII files. You will typically 'read' an ASCII file by instructing *LIMDEP* to locate the file on one of your computer's storage drives and 'read' the file. The next few sections of this chapter will show you how to do that. We begin, however, with a simpler operation that does the same thing. (Section 6.3 also shows you how to read data into *LIMDEP* using the text editor.)

Simple Rectangular ASCII File

The simplest, basic starting point for reading data files, instead of from the text editor, is the rectangular ASCII file that contains exactly one row of variable names at the top of the file and columns of data below it, as shown in the example in Figure 7.4.

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

Figure 7.4 Sample Data File

The sample data set shown illustrates several degrees of flexibility.

- Variable names in a file may be separated by spaces and/or commas.
- 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, tabs, and/or commas.
- 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 *never* understood to be a missing value.)

To read a data file of this form, you need only tell *LIMDEP* where it is. The command to read this file is

READ ; File = … < the name of the file > $

The **READ** command may be submitted from the text editing window or in the command bar at the top of the screen. There are two other ways to import a data file of this form:

- Use Project:Import Variables to open the Import dialog box, as shown in Figure 7.5. Select All Files (*.*) in the file types, then locate and select your data file and click Open.
- In the data editor, select Import Variables from the menu invoked by clicking the right mouse button. (See Figure 7.1.) This opens the same Import dialog box described above.

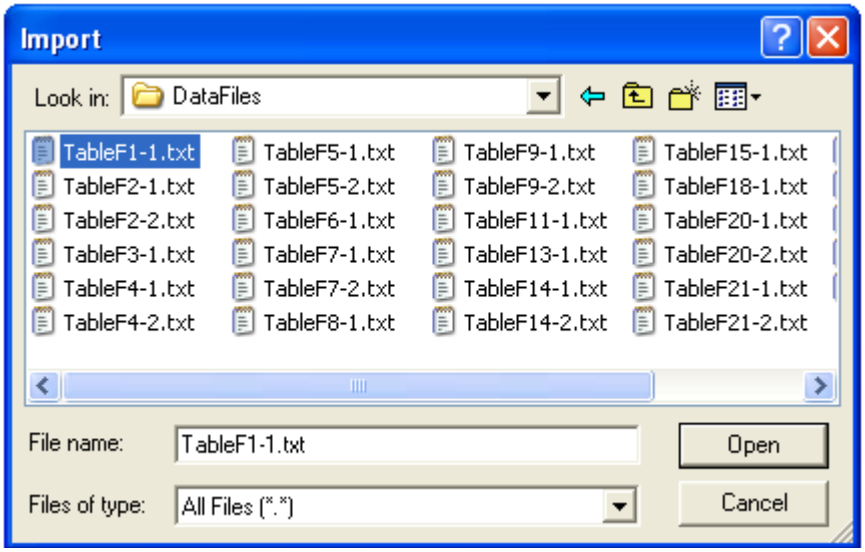


Figure 7.5 Import Dialog Box

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.

Obtaining the Path to a File

The preceding application is one of several situations in which you will 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. For example, where is the file "TableF1-1.txt" that is selected in Figure 7.5? Step one is to make sure that your text editing window is active. The file path will be inserted where the cursor is. Insert:File Path brings up exactly the dialog box shown in Figure 7.5 except that the banner title will be Insert File Path instead of Import. When you click Open, the full path to the file will be placed in double quotes in the text editor. The result is shown in Figure 7.6.

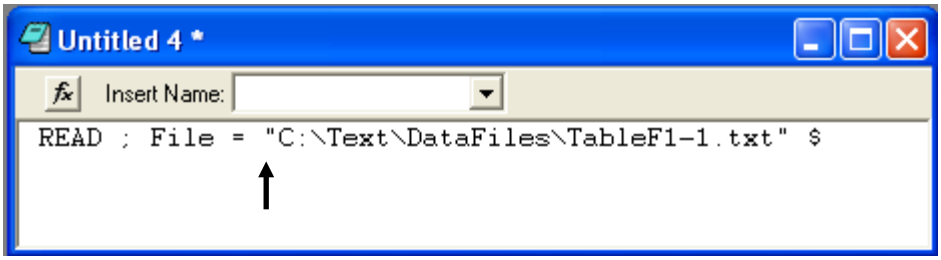


Figure 7.6 Editing Window with Insert File Path

7.2.4 General ASCII Files

The more general command for reading an ASCII data file is

```
READ      ; Nvar  = number of variables
          ; Nobs  = number of observations
          ; Names = names for the variables
          ; File  = the full file name, including path  $
```

(Note that the preceding shows the 'default' form of this command.) This assumes that the file is an ASCII file (not a spreadsheet) with numbers arranged in rows, separated by blanks, tabs, 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 as lines as needed for each observation to supply all of the values. For example, for the following data in a file,

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

you might use

```
READ      ; File = \Project\SMALL.DAT ; Nobs = 4 ; Nvar = 3 ; Names = x,y,z $
```

There are several optional features, and a number of other different types of data files you can use.

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.

Variable Names

The normal way to enter variable names is in the command, as in the example above,

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

The following name conventions apply: 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 4.3.

Names in the Data File

You may find it convenient to make the variable names a part of the data set instead of the **READ** command. (Note that this is the basic case described in the tutorial in the previous chapter.) To do so, add the specification

; Names = *n*

to your **READ** command, where *n* is the number of lines you need to list the names. Then, at the absolute beginning of the data set, include exactly *n* lines of 80 or fewer characters containing the variable names, separated by any number of spaces and/or commas. (For appearance's sake, you might, for example, position the names above the variables.)

The following reads a data set containing three observations on four variables:

READ ; Nobs = 3 ; Nvar = 4 ; Names = 2 ; File ="... MACRO.DAT "$

The data set could be

Year	Consmptn	GNP	Govt
1961	831.25	996.19	128.37
1962	866.95	1024.82	138.83
1963	904.44	1041.03	153.21

Observation Labels

The data file below contains labels for the individual observations as well as the names of the variables. You may read a file with observation labels with the following format:

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

The labels column is an extra column in the data. It is not a variable. For the file below, you would use

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

This indicates that the labels are in the first column, which is probably typical.

```
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)
```

You must include a name for the labels. Note that the 'State' name is used for the labels, but not for the data.

7.2.5 Reading a Spreadsheet File

You can read worksheet files such as those created *by Lotus* (.WKx) or Microsoft *Excel* (.XLS) directly into *LIMDEP* without conversion. The command is simply

READ ; File = name ; Format = WKS $
or **READ ; File = name ; Format = XLS $**

All of the information needed to set up the data is contained within the file. However, if the columns in your worksheet were created by formulas within the spreadsheet program, rather than being typed in initially, then *LIMDEP* will not find the transformed data. The reason is that the .XLS file does not contain the data, themselves, but merely the formula for recreating the data. Since we have not replicated the spreadsheet program inside *LIMDEP*, it is not possible to redo the transformation.

NOTE: Please note the caution about Excel 2007 in Section 6.3

7.2.6 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.

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

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 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 *SAS*'s '.' character, the word 'missing,' or any other code you care to use.

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 your **READ** command. For example, if you have a set of names at the top of your data file and you forget to warn the program to expect them, they will be read as missing values rather than as variable names.

After a data set is **READ**, you are given a count of the variables then existing and a summary of any missing values found.

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 data. It is not necessary to put any alphabetic indicator in the cell.

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.

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.

7.3 Computing Transformed Variables

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.

7.3.1 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 ; name = expression ; ... $

You may also enter your command in a dialog box, as shown in Figure 7.7. The dialog box is invoked by selecting Project:New/Variable or by going to the data editor and pressing 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, and the first observation is set to zero. Remaining observations are treated as missing. You may enter an expression for the new or transformed variable in the Expression window.

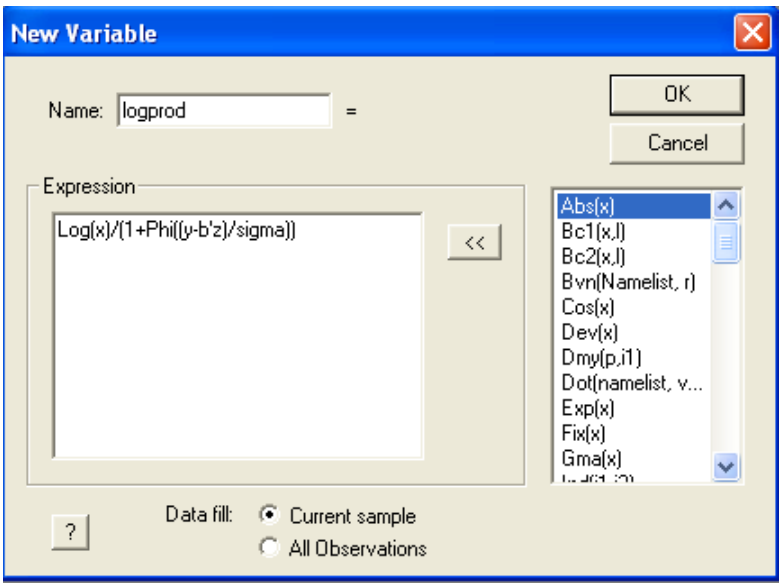


Figure 7.7 New Variable Dialog Box

A **CREATE** command operates on the 'current sample.' (See Section 7.5.) 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 7.7.

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 7.3.3. 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; (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.)	a ! b = max(a,b)
~	= 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 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* $= \dfrac{((x+y)^2-(a+c)^2)^2}{(a+x)(c+y)}$.

You may also nest functions. For a few examples, the following functions are used to invert certain probability distributions:

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)) $
```

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) $

- 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 7.3.3. 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. You should examine the list given in Section 7.3.3.

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. 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.

7.3.2 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... $

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.)

7.3.3 Transformations Involving Missing Values

NOTE: Any mathematical expression that involves a missing value produces a missing value as the result.

Any transformation 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.

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.

7.3.4 CREATE Functions

The expressions in **CREATE** may involve the following functions:

Common Algebraic Functions

Log(x) = natural logarithm,
Exp(x) = exponent,
Abs(x) = absolute value,
Sqr(x) = square root,
Sin(x) = sine,
Rsn(x) = arcsine (operand between -1 and 1),
Cos(x) = cosine,
Rcs(x) = arccosine (operand between -1 and 1),
Tan(x) = tangent,
Ath(x) = hyperbolic arctangent = ½ log((1+x)/(1-x)), -1 < x < 1,
Ati(x) = inverse hyperbolic arctangent = [exp(2x)-1]/[exp(2x)+1],
Gma(x) = gamma function = (x-1)! if x is an 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),
Sgn(x) = sign function = -1,0,1 for x <, =, > 0,
Fix(x) = round to nearest integer,
Int(x) = integer part of operand.

Univariate Normal and logistic Distributions

Phi(x) = CDF of standard normal,
N01(x) = PDF of standard normal,
Lgf(x) = log of standard normal PDF = $-.5(\log 2\pi + x^2) = \text{Log}(\text{N01}(x))$,
Lmm(x) = -N01/Phi = E[x | x < operand], $x \sim N(0,1)$,
Lmp(x) = N01/(1-Phi) = E[x | x > operand],
Lmd(x,z) = (z-1)Lmp(x) - zLmm(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) + zTvp(x) where z = 0/1 (selected variance),
Inp(x) = inverse normal CDF,
Inf(x) = inverse normal PDF (operand is CDF, returns density).
Lgt(x) = logit = log[z/(1-z)],
Lgp(x) = logistic CDF = exp(x)/(1 + exp(x)),
Lgd(x) = logistic density = Lgp(1-Lgp).

Trends and Seasonal Dummy Variables

Trn($x1$,$x2$) = trend = $x1+(i-1)\ x2$ where i = observation number,
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 R6.

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, a familiar transformation, 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) $
```

Matrix Function

A transformations based on matrix algebra is used to create linear forms with the data. Linear combinations of variables are obtained with

```
CREATE      ; name = b'x $
```

where *x* is a namelist of variables (see Section 7.5) 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.

7.3.5 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.
```

Things to note:

- 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.

7.3.6 Random Number Generators

There are numerous transformations which draw samples using *LIMDEP*'s random number generators. The basic generator is the one which will draw a sample from a continuous uniform distribution in the indicated range

```
CREATE      ; name = Rnu (lower limit, upper limit) $
```

and the one which will create a variable containing a sample from the indicated normal distribution,

```
CREATE      ; name = Rnn (mean, standard deviation) $
```

The sample is placed with the observations in the current sample. (Note how we used this feature in the examples in Sections 3.3 and 3.4.)

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 $x1$ and $x2$ 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 can generate random samples from continuous and discrete.

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.

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*.

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. 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.

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 ) ) $
```

Setting the Seed for the Random Number Generator

To reset the seed for the random number generator (the same one is used for all distributions), 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. Using this device will allow you to draw the same string of pseudo-random numbers more than once.

7.4 Lists of Variables

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. The **NAMELIST** feature is used as follows:

NAMELIST ; **name = the list of variables names $**

7.4.1 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**. For example, nearly all model commands are of the form

MODEL COMMAND ; Lhs = a variable
; Rhs = a list of variables
; Rh2 = a list of variables $

Each of the lists may, in principle, have 150 or more names in it. As such, some shorthands will be essential.

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*.

7.4.2 Namelists

The wildcard character described above will save some typing. But, namelists will usually be a more efficient approach to specifying a list of names. The **NAMELIST** command is used to define a single name which will be synonymous with a group of variables. It can be used at any time and applies to the entire set of variables currently in the data array, regardless of how they got there. (Variables are placed in the data array with many commands including **READ, CREATE, MATRIX**, and any of the model commands.) The form of the command 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 $

The lists of variables defined by separate namelists may have names in common. For example,

NAMELIST ; w1 = x1,x2 ; w2 = x2,x3 $

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 7.8 for the setup.

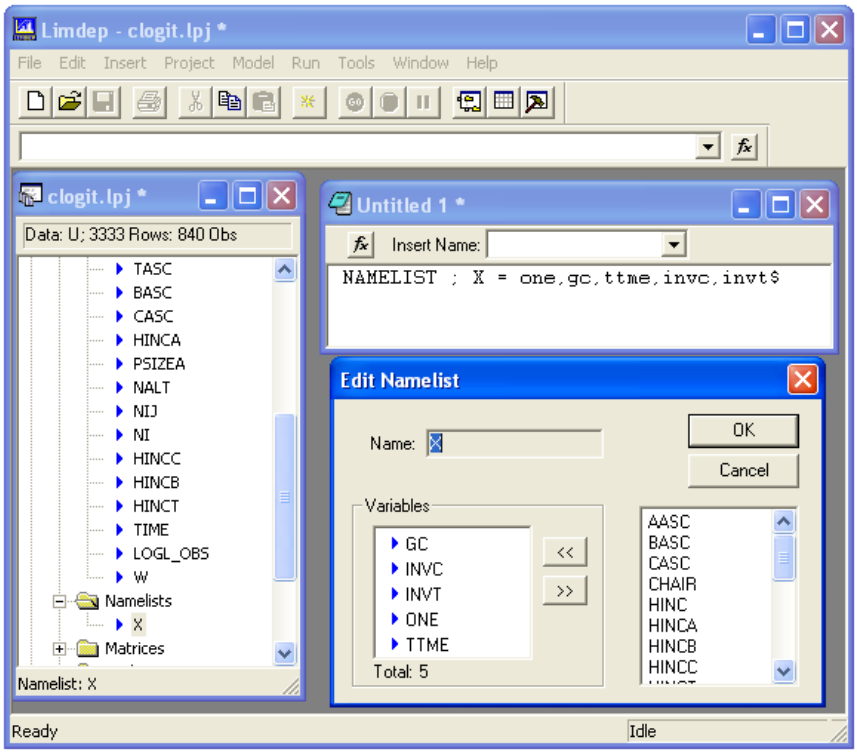


Figure 7.8 Editing a Namelist

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 these will invoke the dialog box shown in Figure 7.9.

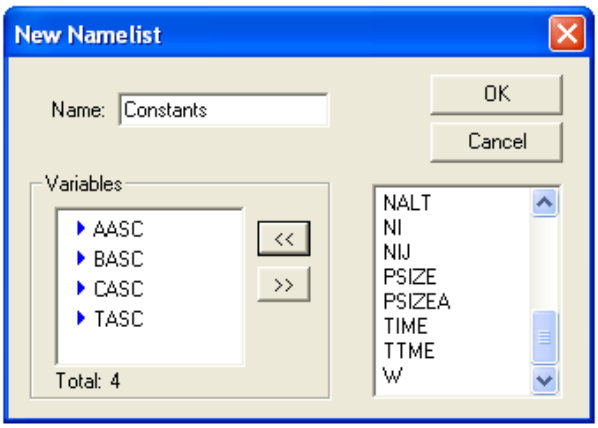


Figure 7.9 New Namelist Dialog Box

7.4.3 Using Namelists

Namelists will have many uses as you use *LIMDEP* to analyze your data. Consider the 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*.

Your primary use of the **NAMELIST** command will be for defining variable lists for the estimation commands. However, a namelist may be used at any point at which a list of variable names is called for. Some other applications are in defining data matrices for the **MATRIX** command and in several **CALCULATE** and **CREATE** commands. Some examples:

```
NAMELIST  ; xlist = x1, x2, x3, x4 $
REGRESS   ; Lhs = y ; Rhs = one, xlist $
CALC      ; Col(xlist) $   How many variables in list?
WRITE     ; xlist, y1, y2, y3 ; File = DATA.DAT $
MATRIX    ; xtx = <x'x> $   Computes an (X′X)^-1 matrix.
```

7.5 The Current Sample of Observations

In many cases, you will simply analyze the entire data set that you input. But, most analyses also involve partitioning the data set into subsamples, either by stratification, or by excluding or including observations based on some data related criteria. This section will describe these two aspects of operation. The final section will include a discussion of how the sample is modified, either by you or automatically, when the data set contains missing observations.

SAMPLE	designate specific observations to be included in a subsample,
DATES	establish the periodicity of time series data,
PERIOD	designate specific time series observations to be included in a subsample,
REJECT	exclude certain observations from the sample based on an algebraic rule,
INCLUDE	include certain observations from the sample based on an algebraic rule,

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 section 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 a simple **SAMPLE ; All $**. Figure 7.10 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.

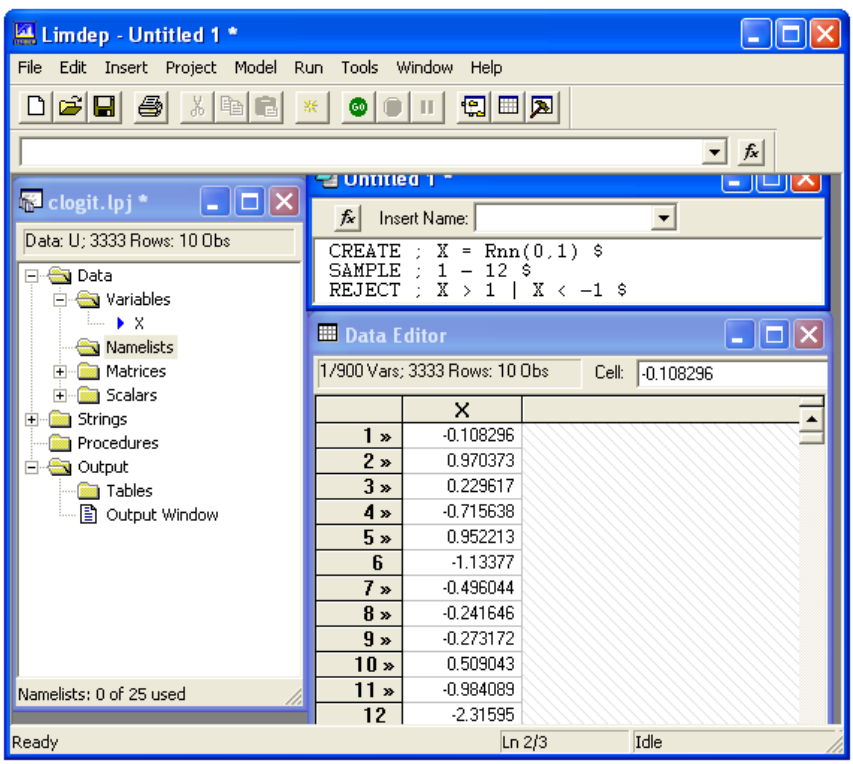


Figure 7.10 Current Sample and the REJECT Command

7.5.1 Cross Section Data

Initially, observations are defined with respect to 'rows' of the data matrix, which are simply numbered 1 to 1,000.

Sample Definition – 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 $**.

Exclusion and Inclusion – The REJECT and INCLUDE Commands

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 7.10. 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 $

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, this has the result of first setting the sample to All, then rejecting all 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*.

You may enter **REJECT** and **INCLUDE** commands from the dialog box shown in Figure 7.11. 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 'Add observations to 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.

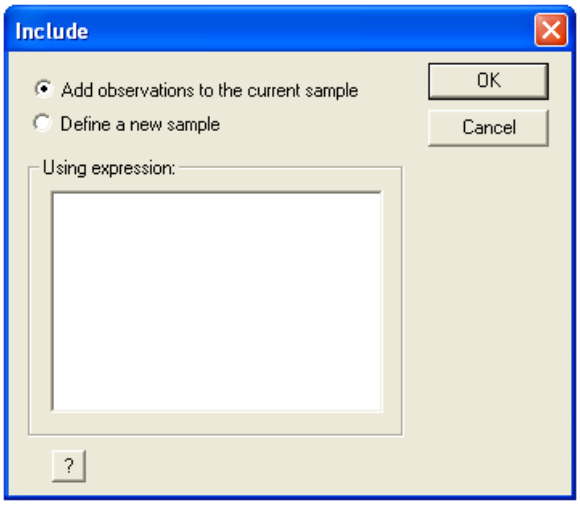


Figure 7.11 INCLUDE and REJECT Dialog Box

The same Set Sample menu offers All, which just generates a **SAMPLE ; All $** command and Range which produces the dialog box shown in Figure 7.12.

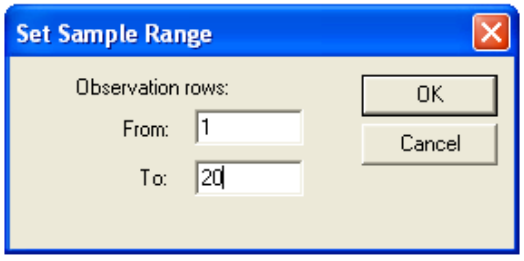


Figure 7.12 Set Sample Range Dialog Box

TIP: If your **REJECT** 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.

Interaction of REJECT/INCLUDE and SAMPLE

REJECT and **INCLUDE** modify 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. The second **SAMPLE** command undoes then replaces the first one.

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. This is generally not done automatically.

7.5.2 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 menu item.

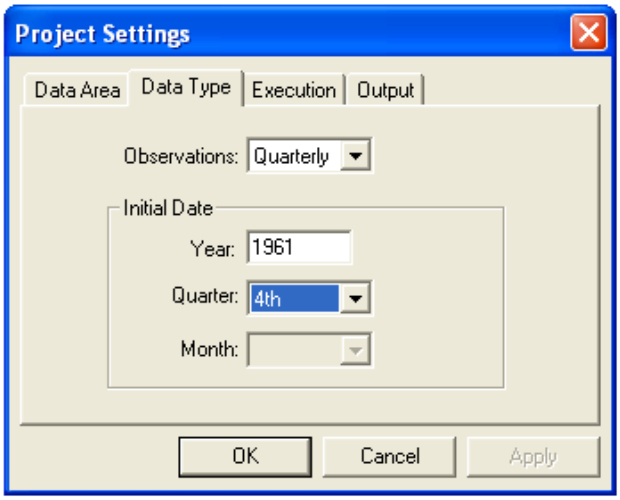


Figure 7.13 Dialog Box for DATES Command

The **SAMPLE** and **PERIOD** commands may be given from the Project:Set Sample Range dialog box. See Figure 7.14.

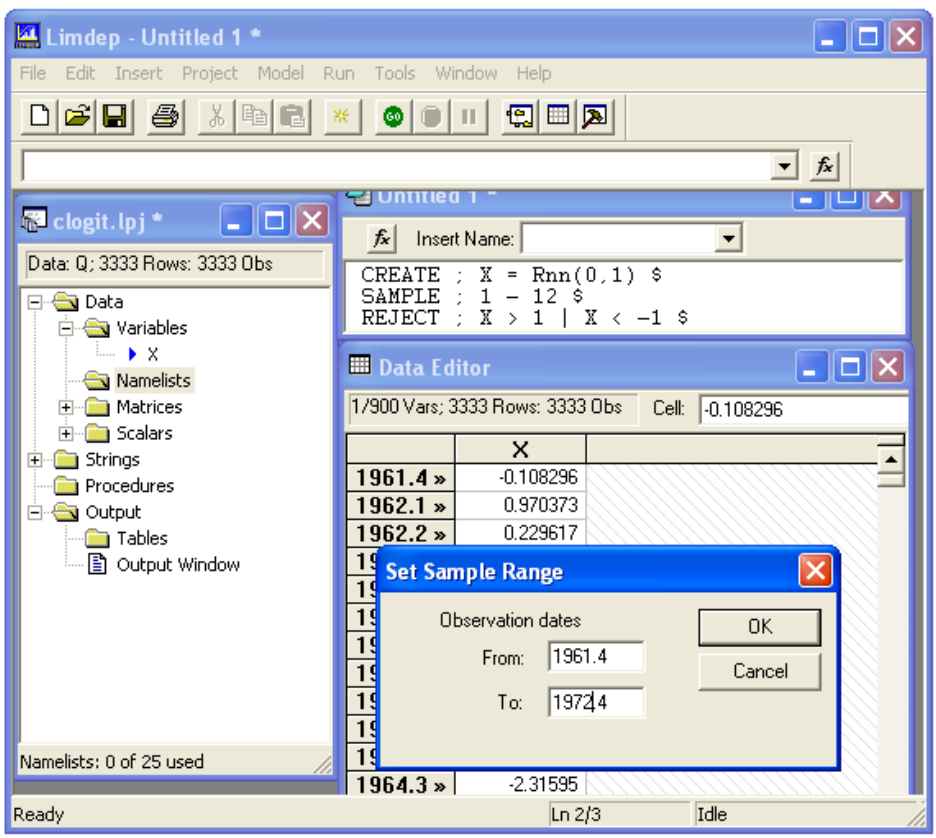


Figure 7.14 Dialog Box for the PERIOD Command

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 7.13 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 7.14 for quarterly data beginning in 1961.4.

7.6 Missing Data

Observations that contain missing values for variables in a model are automatically bypassed. (The full version of *LIMDEP* uses a **SKIP** command for this setting. The **SKIP** switch is automatically turned on in the Student version.) Missing values are handled specifically elsewhere in the program by the **CREATE**, **CALCULATE** and **MATRIX** commands.

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' during matrix computations (the student version does), *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 7.15 illustrates the results discussed in the previous paragraph. The commands are shown in the editing window. 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 88 observations in the reduced sample (see the 88 that is **1′1** at the upper left corner) and a warning is issued. With **SKIP** turned off, in the second computation, the missing values are treated as -999s, and the resulting matrix has values that appear to be inappropriate.

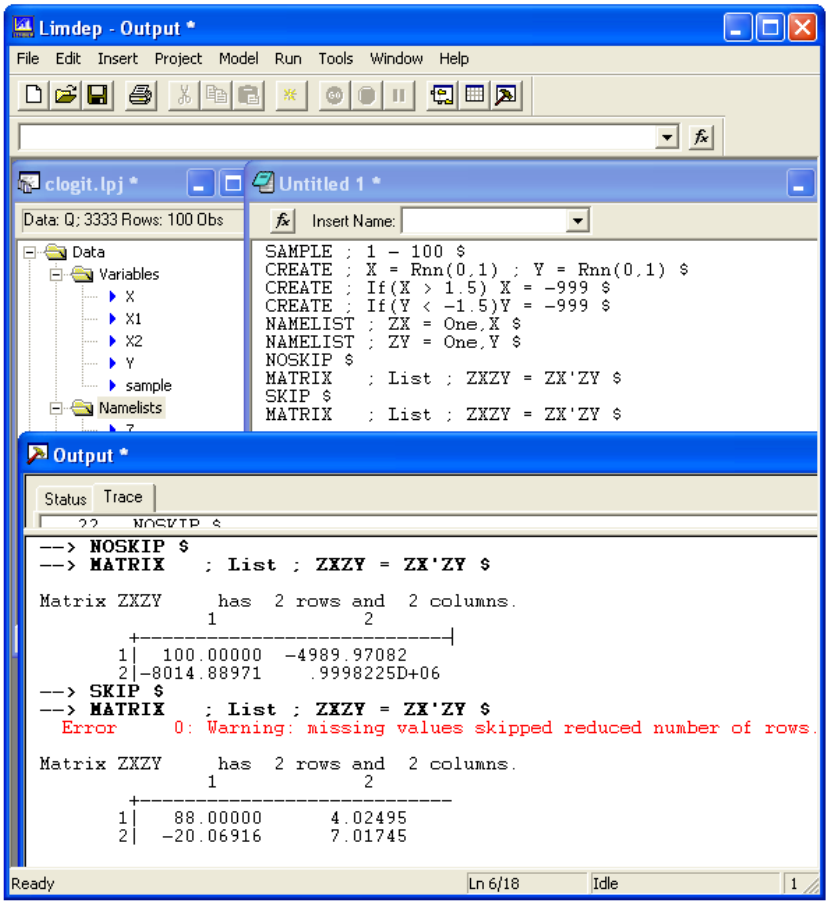


Figure 7.15 Matrix Computations Involving Missing Data

Chapter 8: Estimating Models

8.1 Introduction

Once your data are in place and you have set your desired current sample, most of your remaining commands will be either model estimation commands or the data manipulation commands, **CREATE**, **CALCULATE**, and **MATRIX**. We consider the model commands in this chapter and the other commands in Chapter 9. This chapter will describe the common form of all model estimation commands, estimation results, and how to produce useable output in an output file. Section 8.3 contains a general discussion on the important statistical features of the model estimators, such as using weights and interpreting resultsinal effects. This chapter will 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, this chapter is the most important general chapter in this part of the manual. The chapters in the second part of this manual will provide free standing and fairly complete descriptions of how to estimate specific models, but users are encouraged to examine Chapters 8 and 9 here closely for the essential background on these procedures.

8.2 Model Estimation Commands

Nearly all model commands are variants of the basic structure

MODEL COMMAND ; Lhs = dependent variable
; Rhs = list of independent variables
; ... other parts specific to the model ; ... $

The 75 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. Different models will usually require different numbers and types of variables to be specified in the lists above. Note that, in general, 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 75 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.*

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 the 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 remove it from the list.)

NOTE: With all of the different forms and permutations of features, *LIMDEP* supports several hundred different models. They are described fully in the 3 volume documentation for the full program. This student version contains all of those models, however, the manual will only describe a few of the available model specifications.

In addition to the specifications of variables, there are roughly 150 different specifications of the form

; sss [= additional information]

which are used to complete the model command. In some cases, these are mandatory, as in

REGRESS ; Lhs = ... ; Rhs = ... ; Panel ; Str = variable $.

This is the model command for the fixed and random effects linear regression model. The latter two specifications are necessary in order to request the panel data model. Without them, the command simply requests linear least squares. 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*.

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.

8.2.1 The Command Builder

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 directly as shown above. Figure 8.1 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.

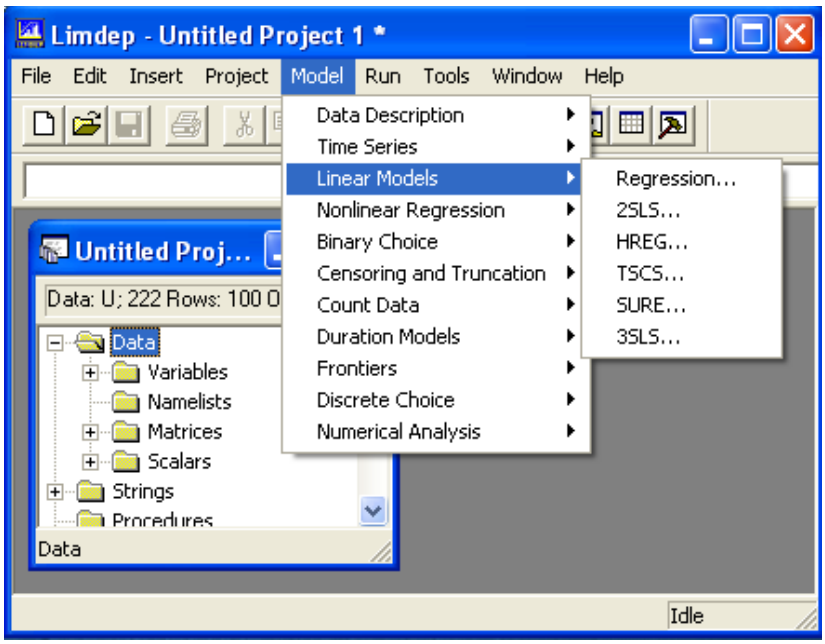


Figure 8.1 Model Command Builder with Linear Models Menu

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 8.2. The variables, *d*1, *d*2,… are dummy variables for the 10 firms.

Figure 8.3 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 Section 8.3.) 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.

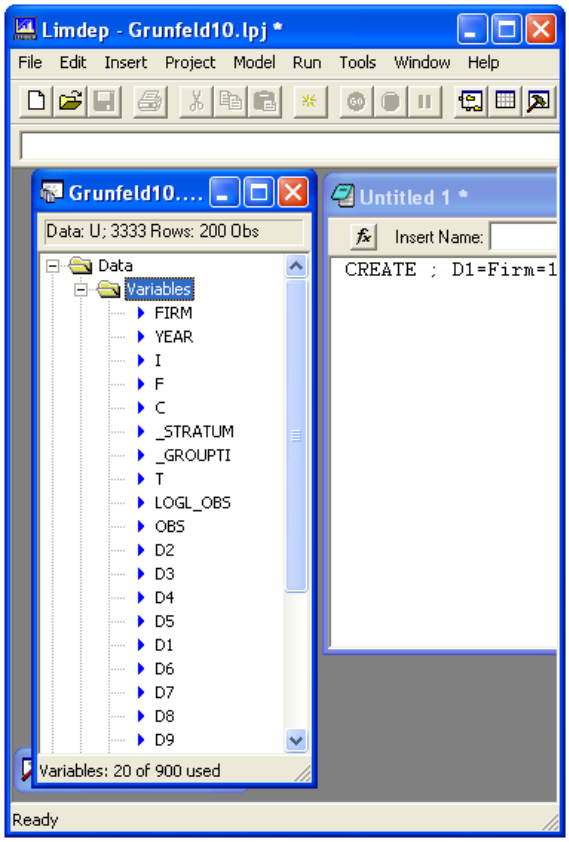


Figure 8.2 Project Window for the Grunfeld Data

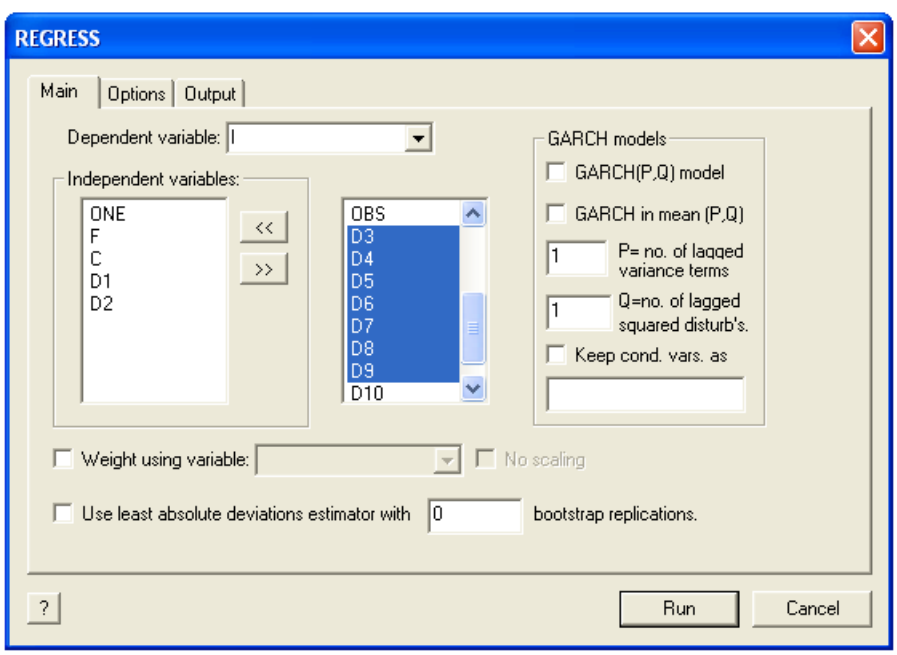


Figure 8.3 Main Page of Command Builder for Linear Regression Model

The other two tabs in the command builder provide additional options for the linear regression model, as shown in Figures 8.4 and 8.5. 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 8.6.

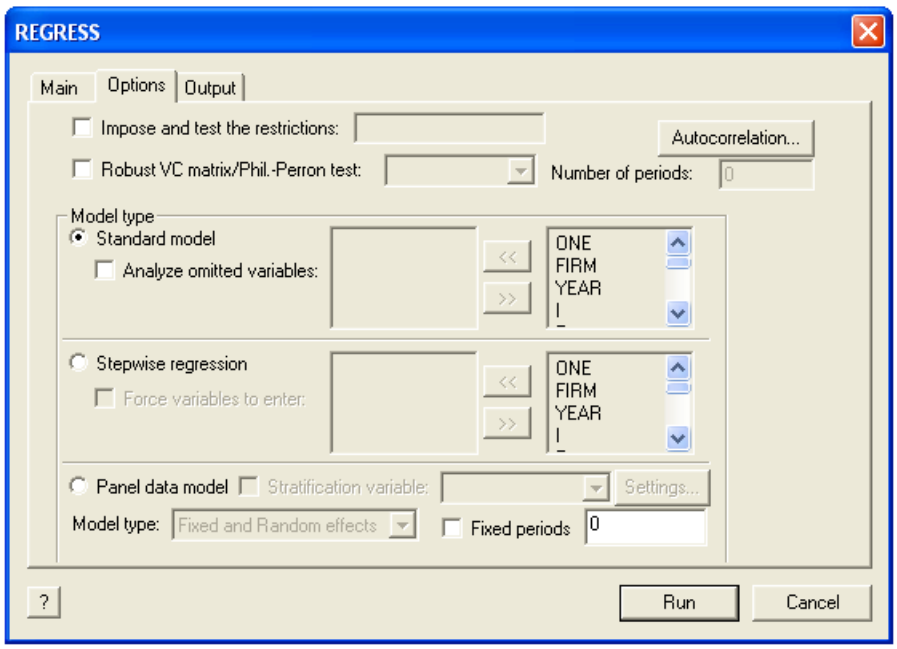


Figure 8.4 Options Page for Linear Regression Model

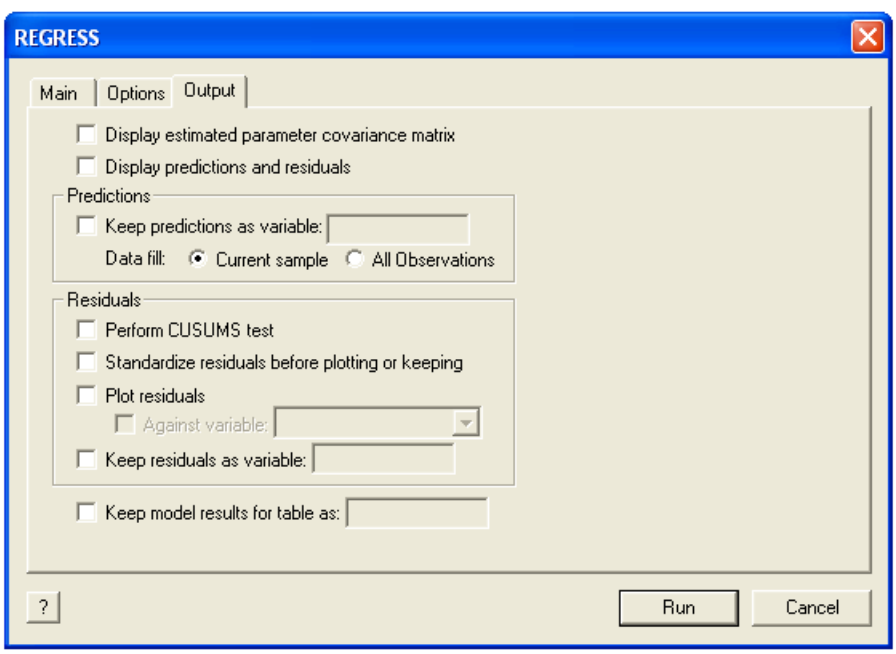


Figure 8.5 Output Page for Linear Regression Model

The regression command that was assembled by the command builder can be seen in the output window directly above the regression output:

REGRESS ; Lhs = i ; Rhs = one,f,c,d1,d2,d3,d4 $

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/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 Rhs 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.

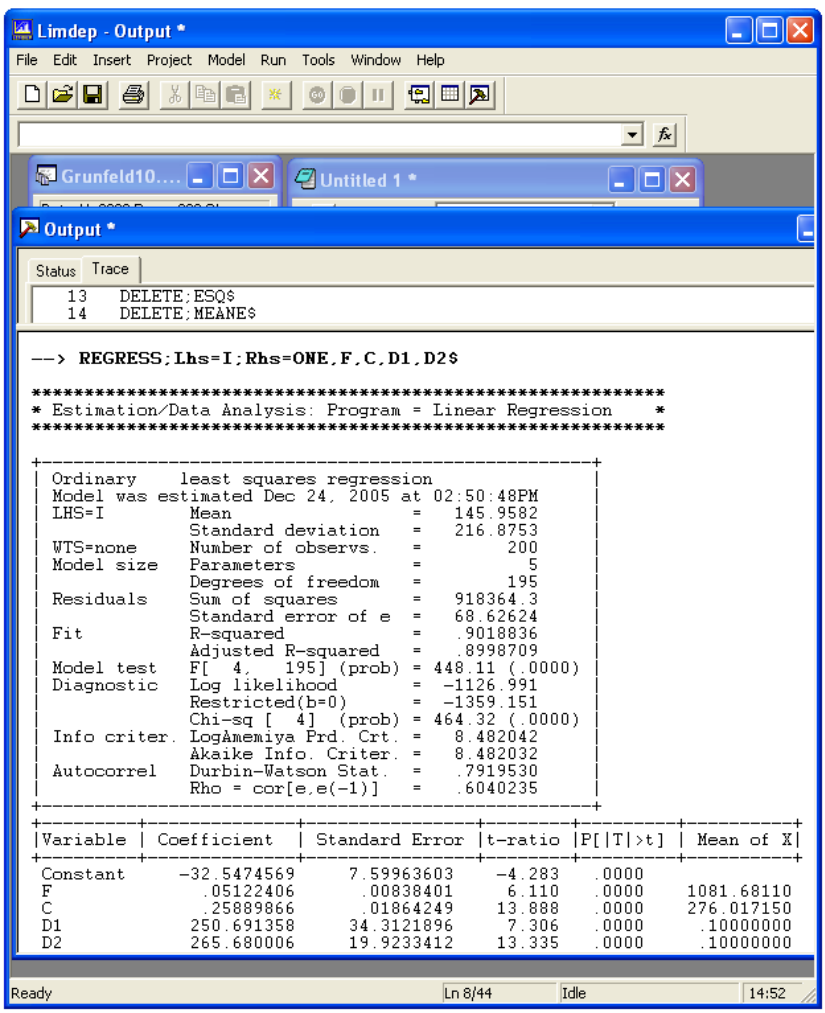


Figure 8.6 Regression Output Window

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.

8.2.2 Output from Estimation Programs

Results produced by the estimation commands will, of course, vary from model to model. The display in the output window in Figure 8.7 would be typical. These are the results produced by estimation of a basic tobit model. (We converted the dependent variable in the preceding linear model to deviations from the overall mean then forced a tobit model on the resulting variable.)

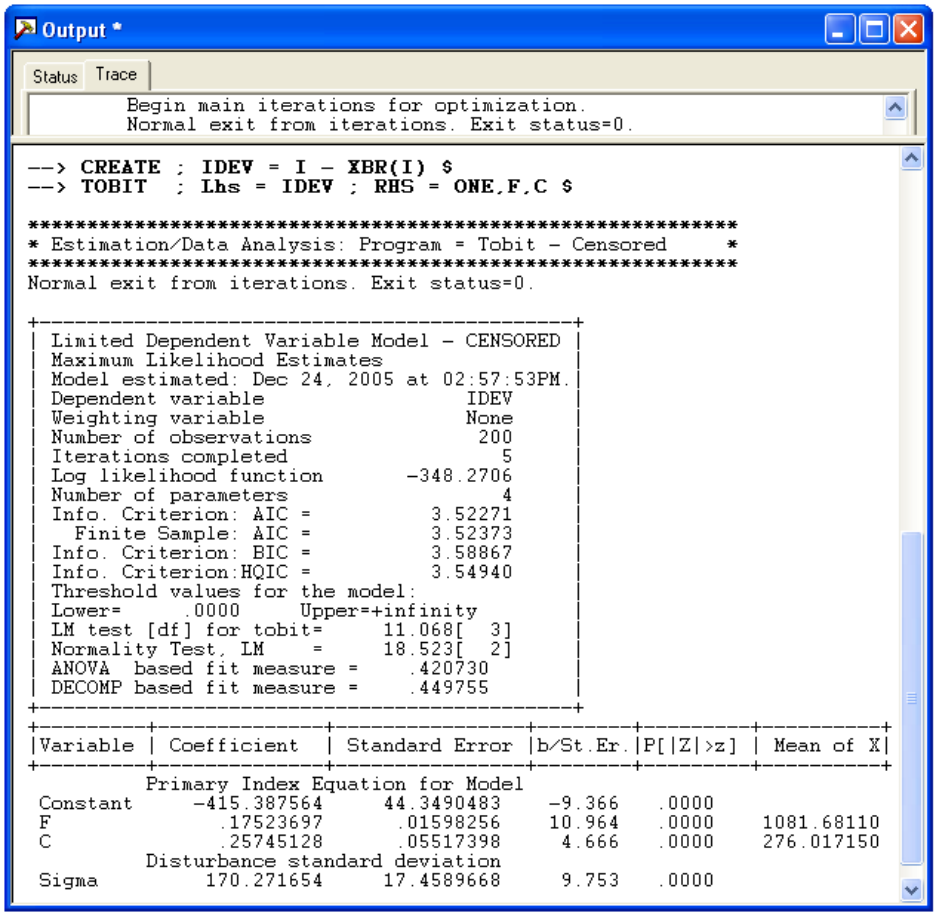


Figure 8.7 Output Window for an Estimated Model

Displaying Covariance Matrices

One conspicuous absence from the output display 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

; Printvc

to the model command. Since covariance matrices can be extremely large, this is handled two ways. If the resultant matrix is quite small, it is included in the output. The earlier tobit equation is shown in Figure 8.8. If the matrix has more than five columns, then it is offered as an additional embedded matrix with the output, as shown in Figure 8.9 for a larger regression model. (We also included **; Matrix** in this estimation. Note that there are two embedded matrices in the output.)

When estimation is done in stages **; Printvc** will only produce an estimated covariance matrix at the final step. Thus, no covariance matrix is displayed for initial least squares results.

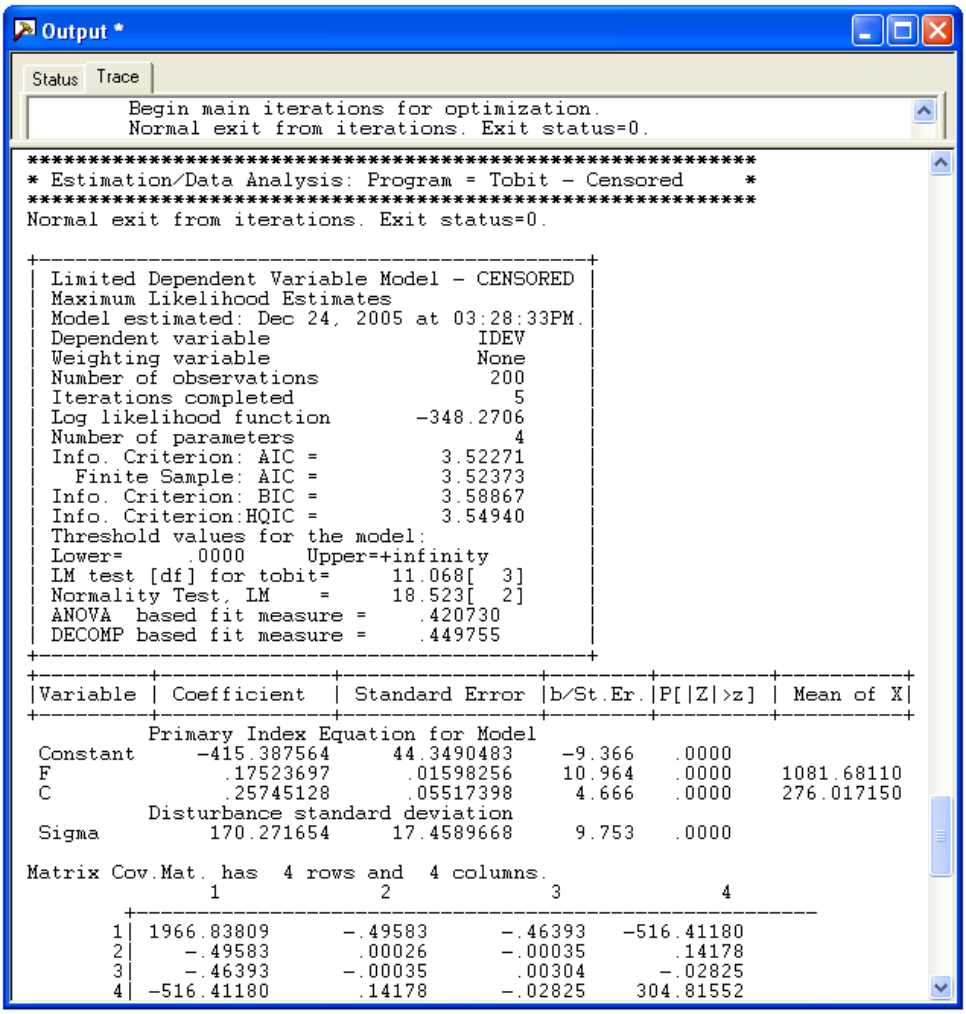


Figure 8.8 Model Output with Covariance Matrix

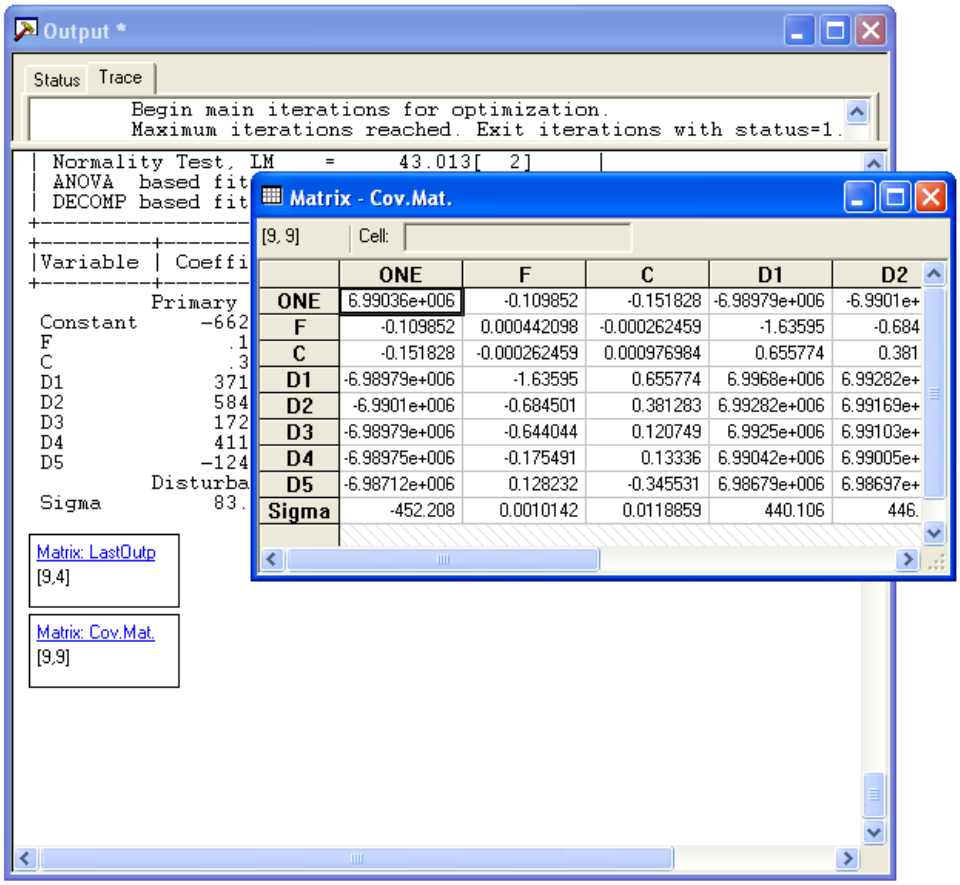


Figure 8.9 Regression Output with Embedded Covariance Matrix

8.3 Model Components and Results

The primary components of a model are provided with the model command

MODEL COMMAND ; Lhs = dependent variable
; Rhs = independent variables $

In many cases, this is all that is required. However there are a few common variations, notably using weights.

Constant Terms

As noted earlier, *LIMDEP* almost never forces a constant term in a model – the only case is the linear regression model fit by stepwise least squares. For any other case, if you want a constant term in a model, you must include the variable *one* in the appropriate variable list. 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.

The typical estimation command will appear like the following example:

REGRESS ; Lhs = y ; Rhs = one, x $

which produces output including

```
+---------+--------------+----------------+--------+---------+----------+
|Variable | Coefficient  | Standard Error |b/St.Er.|P[|Z|>z] | Mean of X|
+---------+--------------+----------------+--------+---------+----------+
 Constant -.00280548254    .0070621362      -.397   .6912
 X         .01489733909    .0071169806      2.093   .0363    -.006473082
```

A **MATRIX** command based on the same construction might be the following which computes the inverse of a matrix of sums of squares and cross products:

NAMELIST ; xdata = one,x $
MATRIX ; xxi = <x'x> $

8.3.1 Using Weights

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 $\text{Log } L = \Sigma_i w_i \log(f_i)$, 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.

8.3.2 Model Output

Most of the models estimated by *LIMDEP* are single equation, 'index function' models. That is, 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, θ, such as a variance term, σ^2 in a regression or a tobit model. The parameters to be estimated are $[\boldsymbol{\beta},\theta]$. In this framework, estimation will usually begin with a least squares regression of y on x. This will often be for the purpose of obtaining the default starting values for the iterations, but is sometimes done simply because in this modeling framework, the ordinary least squares (OLS) estimate is often an interesting entity in its own right. Model output, not including the technical output from the iterations (see Section R8.4) will thus consist of the OLS results, followed by the primary objects of estimation.

NOTE: In order to reduce the amount of superfluous output, OLS results are not reported automatically except for the linear regression model. To see the OLS outputs when they are computed, add **; OLS** to your model command.

TIP: It is important to keep in mind that the OLS estimates are almost never consistent estimates of the parameters of the nonlinear models estimated by *LIMDEP*.

Listed below are the results from estimation of a Poisson regression model. (The example is used at various points in the chapters to follow.) The model command requests a set of marginal effects. These are discussed in the next section.

```
READ       ; Nobs = 40 ; Nvar = 2 ; Names = num,months ; By Variables $
             0 0 3 4  6 18  m 11 39 29 58 53 12 44  m 18  1  1  0 1
             6 2 m 1  0  0  0  0  2 11  m  4  0  m  7  7  5 12  m 1
             127 63 1095 1095 1512 3353 0 2244 44882 17176 28609 20370
             7064 13099 0 7117 1179 552 781 676 783 1948 0 274 251 105
             288 192 349 1208 0 2051 45 0 789 437 1157 2161 0 542
REJECT     ; num < 0 $
CREATE     ; a = Ind(9,16) ; c = Ind(17,24) ; d = Ind(25,32) ; e = Ind(33,40)
           ; c67 = Dmy(8,3)+Dmy(8,4) ; c72  = Dmy(8,5)+Dmy(8,6)
           ; c77 = Dmy(8,7)+Dmy(8,8) ; p77 = Dmy(2,2)
           ; logmth = Log(months) $
NAMELIST   ; x = one,a,c,d,e,c67,c72,c77,logmth$
POISSON    ; Lhs = num ; Rhs = x ; OLS ; Marginal = c67 $
```

The following results are reported when your model command contains **; OLS**.

```
+-----------------------------------------------------+
| Poisson Regression Model - OLS Results              |
| Ordinary    least squares regression                |
| LHS=NUM      Mean                 =   10.47059      |
|              Standard deviation   =   15.73499      |
| WTS=none     Number of observs.   =         34      |
| Model size   Parameters           =          9      |
|              Degrees of freedom   =         25      |
| Residuals    Sum of squares       =   2076.755      |
|              Standard error of e  =   9.114286      |
| Fit          R-squared            =   .7458219      |
|              Adjusted R-squared   =   .6644848      |
+-----------------------------------------------------+
+---------+--------------+----------------+--------+---------+----------+
|Variable | Coefficient  | Standard Error |b/St.Er.|P[|Z|>z] | Mean of X|
+---------+--------------+----------------+--------+---------+----------+
 Constant     -39.6343740     9.30415297    -4.260   .0000
 A             4.38284585     5.12356623      .855   .3923     .23529412
 C            -1.60558349     4.85887959     -.330   .7411     .23529412
 D             1.75161158     4.99174267      .351   .7257     .23529412
 E             1.28368407     7.62716506      .168   .8663     .05882353
 C67            .13630013     4.67310647      .029   .9767     .29411765
 C72          -5.06717068     4.97659408    -1.018   .3086     .29411765
 C77          -7.29195392     5.63651443    -1.294   .1958     .14705882
 LOGMTH        7.30381292     1.36910416     5.335   .0000    7.04925451
```

The following are the standard results for a model estimated by maximum likelihood or some other technique. Most parts are common, while a few are specific to the particular model.

```
+--------------------------------------------+
| Poisson Regression                         |
| Maximum Likelihood Estimates               |
| Dependent variable                  NUM    |
| Number of observations               34    |
| Log likelihood function       -72.69771    |
| Number of parameters                  9    |
| Info. Criterion: AIC =          4.80575    |
|   Finite Sample: AIC =          5.02634    |
| Info. Criterion: BIC =          5.20978    |
| Info. Criterion:HQIC =          4.94354    |
| Restricted log likelihood     -356.2029    |
| Chi squared                    567.0104    |
| Degrees of freedom                    8    |
| Prob[ChiSqd > value] =         .0000000    |
| Chi- squared =    46.55065  RsqP=   .9403  |
| G  - squared =    47.52893  RsqD=   .9227  |
| Overdispersion tests: g=mu(i)  :  1.770    |
| Overdispersion tests: g=mu(i)^2:   .498    |
+--------------------------------------------+
+---------+-------------+----------------+--------+---------+----------+
|Variable | Coefficient | Standard Error |b/St.Er.|P[|Z|>z] | Mean of X|
+---------+-------------+----------------+--------+---------+----------+
 Constant     -4.80807969      .65751784    -7.312   .0000
 A              .00373503      .15767690      .024   .9811     .23529412
 C             -.83536379      .36686744    -2.277   .0228     .23529412
 D              .29514731      .24430603     1.208   .2270     .23529412
 E              .13092973      .32043868      .409   .6828     .05882353
 C67            .70947001      .17792942     3.987   .0001     .29411765
 C72            .86911092      .19268330     4.511   .0000     .29411765
 C77            .51343007      .24440891     2.101   .0357     .14705882
 LOGMTH         .80295174      .06534072    12.289   .0000    7.04925451
```

Some notes about the results shown above:

- The sample was changed from the initial 40 observations to the 34 observations with a nonmissing value of *num* before estimation.
- The least squares results are the standard set of results that will appear with every ordinary least squares regression. However, the table of diagnostic statistics shown above the coefficient estimates is slightly abridged when the model command is not specifically for a linear regression. The linear regression model would show autocorrelation diagnostics and other information criteria.
- 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 few others), the parameter label that you provide will appear with the estimate instead.
- Results for index function models include the mean values of the variables that are multiplied by the coefficients. Ancillary parameters and user defined parameters will show blanks in this part of the table. Note that ‘one’ becomes ‘Constant’ in the table.

- The prob value shown, '$P[|Z| > z]$,' 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.

- The diagnostics table for the Poisson regression reports some statistics which will be present for all models (indicated by the box in the figure):

    1. left hand side variable,
    2. number of observations used,
    3. number of iterations completed,
    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. In this example, chi squared, *G* squared and two overdispersion tests are statistics that are specific to the Poisson model.

8.3.3 Retrievable Results

When you estimate a model, the estimation results are displayed on the screen and in the output file if one is open. In addition, each model produces a number of results which are saved automatically and can be used in subsequent procedures and commands. The **POISSON** 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)) $
```

to compute a standard *F* statistic. (There is an easier way to do this.)

Although your **CALCULATOR** has 50 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 8.10 illustrates. This shows the setup of the project window after the tobit 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**.

All estimators set at least some of these matrices and scalars. In the case of the scalars, those not saved by the estimator are set to 0. 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.

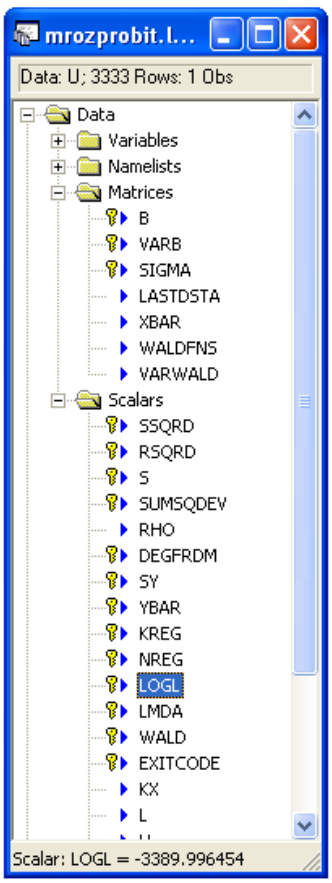


Figure 8.10 Project Window

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

$$\begin{bmatrix}\beta\\ \sigma\end{bmatrix} \text{ and } \begin{bmatrix}\mathbf{V}_{\beta\beta} & \mathbf{V}_{\beta\sigma}\\ \mathbf{V}_{\sigma\beta} & \mathbf{V}_{\sigma\sigma}\end{bmatrix}.$$

The results kept are β, 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 the previous section.) 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.

8.3.4 Creating and Displaying Predictions and Residuals

Most 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 a bit ambiguous, but, once again, some construction that typically reflects a deviation of an actual from a predicted value can usually be retained. The exact definition of a 'fitted value' and a 'residual' are given with the model descriptions in the chapters to follow.

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 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 specification

; List

requests a listing of the residuals and several other variables.

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 interpolate and produce predicted values for these observations by adding the specification

; Fill

to your model command. **; 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: **; 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.

```
READ        ; Nobs = 27 ; Nvar = 10 ; Names = 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 $
```

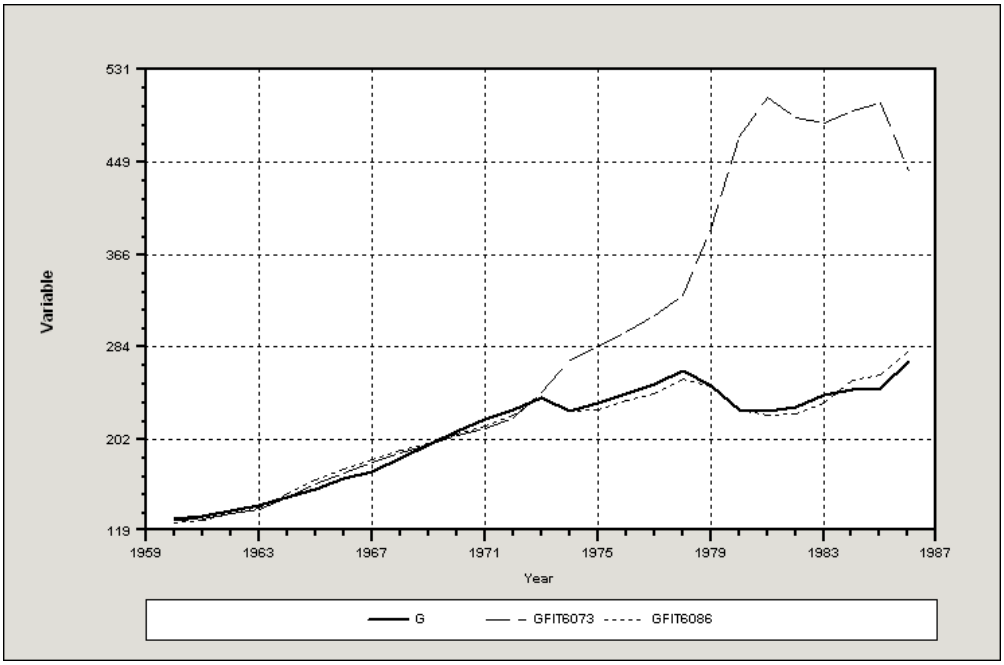


Figure 8.11 Time Series Plot

The following chapters 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:	$E[y] = \exp(\mathbf{b}'\mathbf{x})$,
Residual:	$y - E[y]$,	Index:	$\mathbf{b}'\mathbf{x}$,
Probability:	$\Pr[Y = y] = \exp(-\lambda)\lambda^y/y!$, $\lambda = E[y]$.		

```
LIMDEP Estimation Results                  Run log line    6  Page   3
Current sample contains      34 observations.
Predicted Values    * ⇒ observation was not in estimating sample.
Observation Observed Y   Predicted Y    Residual       x(i)ß    Pr[y*=y]
       1    0.00000       0.39914        -0.3991      -0.9184     0.6709
       2    0.00000       0.22733        -0.2273      -1.4813     0.7967
       3     3.0000        4.5761        -1.5761       1.5209     0.1644
       4     4.0000        4.5761        -0.5761       1.5209     0.1881
       5     6.0000        6.9559        -0.9559       1.9396     0.1499
       6     18.000        13.185         4.8150       2.5791     0.0426
       8     11.000        6.6922         4.3078       1.9009     0.0375
       9     39.000        44.388        -5.3881       3.7930     0.0453
      10     29.000        20.603         8.3967       3.0255     0.0162
```

For a linear regression, the listed items are the familiar ones:

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       129.70        127.98         1.7243     113.5586   142.3929
  1961       131.30        129.47         1.8315     115.8623   143.0746
  1962       137.10        134.80         2.3032     121.5356   148.0580
  1963       141.60        138.04         3.5551     124.9718   151.1179
  1964       148.80        148.58          .2217     134.9251   162.2315
  1965       155.90        160.79        -4.8910     147.7508   173.8312
  1966       164.90        170.39        -5.4906     157.4244   183.3568
  1967       171.00        180.11        -9.1072     167.4143   192.8002
  1968       183.40        187.95        -4.5490     175.1447   200.7533
  1969       195.80        195.73          .0665     182.9210   208.5460
  1970       207.40        204.04         3.3641     191.1278   216.9441
  1971       218.30        210.46         7.8377     197.3139   223.6107
  1972       226.80        219.00         7.8000     205.4731   232.5270
  1973       237.90        242.57        -4.6662     226.7579   258.3745
* 1974       225.80        271.46       -45.6629     197.8524   345.0734
* 1975       232.40        282.81       -50.4150     193.9795   371.6504
* 1976       241.70        295.84       -54.1380     199.1641   392.5119
* 1977       249.20        310.71       -61.5120     201.1378   420.2862
* 1978       261.30        328.38       -67.0848     210.8962   445.8733
* 1979       248.90        388.25      -139.3480     168.3394   608.1566
* 1980       226.80        469.95      -243.1545      93.6745   846.2345
* 1981       225.60        505.76      -280.1606      67.3596   944.1616
* 1982       228.80        486.73      -257.9265      80.0424   893.4107
* 1983       239.60        482.43      -242.8259      97.6831   867.1688
* 1984       244.70        493.02      -248.3244     122.6819   863.3669
* 1985       245.80        499.99      -254.1909     126.6925   873.2893
* 1986       269.40        439.62      -170.2191     191.3035   687.9347
```

Chapter 9: Using Matrix Algebra

9.1 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 defining data matrices with 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 marginal effects or covariance matrices for two step estimators.

To introduce this extensive set of tools and to illustrate its flexibility, we will present two examples. (These are both built in procedures in *LIMDEP*, so the matrix programs are only illustrative.) The rest of the chapter will provide some technical results on matrix algebra and material on how to use **MATRIX** to manipulate matrices.

Example 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 $\mathbf{R}\mathbf{b}^* = \mathbf{q}$ are

$$\mathbf{b}^* = \mathbf{b} - (\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}(\mathbf{R}\mathbf{b}-\mathbf{q}),$$

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 = ... $
CREATE      ; y = the dependent variable $
SAMPLE      ; ... as appropriate  $
```

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 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
         ; Stat(br,vr,x) $
```

The preceding gives the textbook case 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:

$$\begin{bmatrix} \mathbf{X'X} & \mathbf{R'} \\ \mathbf{R} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{b}^* \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{X'y} \\ \mathbf{q} \end{pmatrix}.$$

If the matrix in brackets can be inverted, then the restricted least squares solution is obtained along with the vector of Lagrange multipliers, λ. 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. The routine for this 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) $
```

9.2 Entering MATRIX Commands

MATRIX commands are typically given as parts of programs that perform larger functions, such as in the examples in Section 9.2.1. 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).

9.2.1 The Matrix Calculator

You may invoke the matrix calculator by selecting Tools:Matrix Calculator as shown in Figure 9.1.

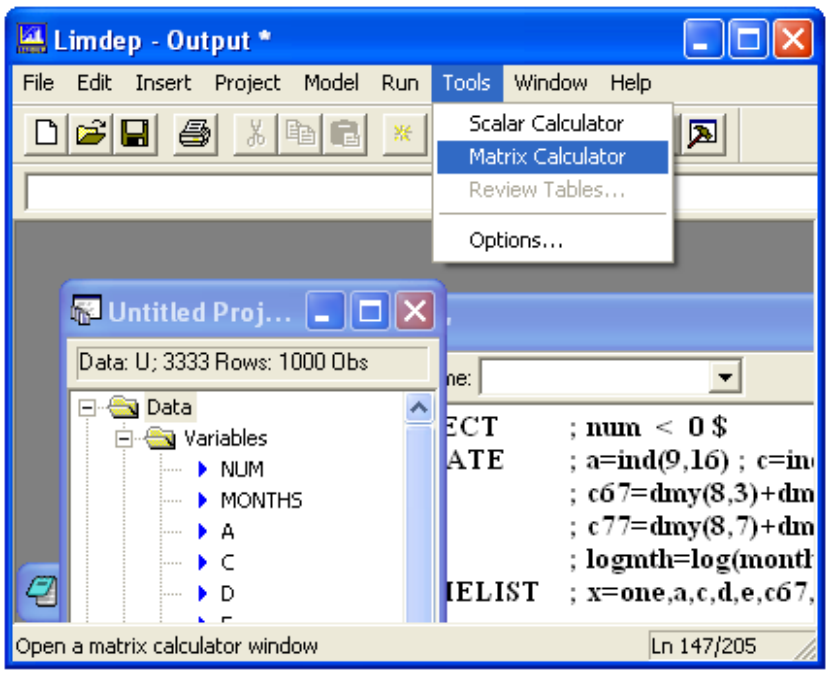


Figure 9.1 Tools Menu for Matrix Calculator

The matrix calculator window is shown in Figure 9.2.

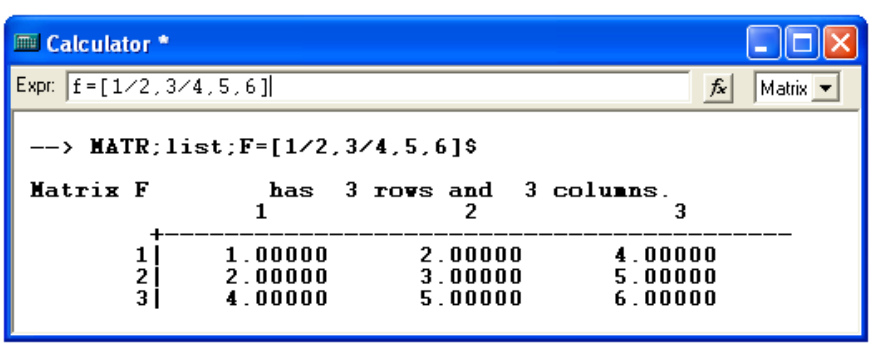


Figure 9.2 Matrix Calculator Window

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 9.3.

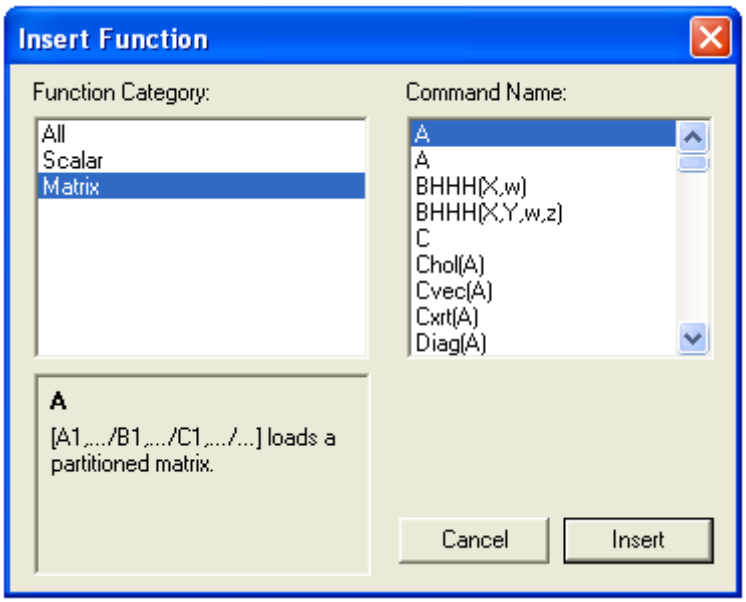


Figure 9.3 Insertion Window for Matrix Functions

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.

9.2.2 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 R12.1 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 =**.' (The same applies to the scalar calculator described in the next chapter.) 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 $

Matrix Result   has  4 rows and  4 columns.
               1            2            3            4
 +-------------------------------------------------------
1|    1.00000       .66647       .13895       .70138
2|     .66647      1.00000      -.27401       .25449
3|     .13895      -.27401      1.00000      -.05935
4|     .70138       .25449      -.05935       .00100
```

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 9.4.

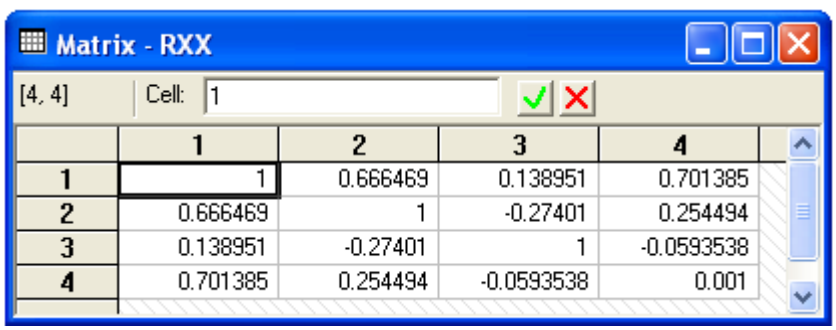


Figure 9.4 Matrix Display from Project Window

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.

9.2.3 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 9.2.5.) The figures below demonstrate.

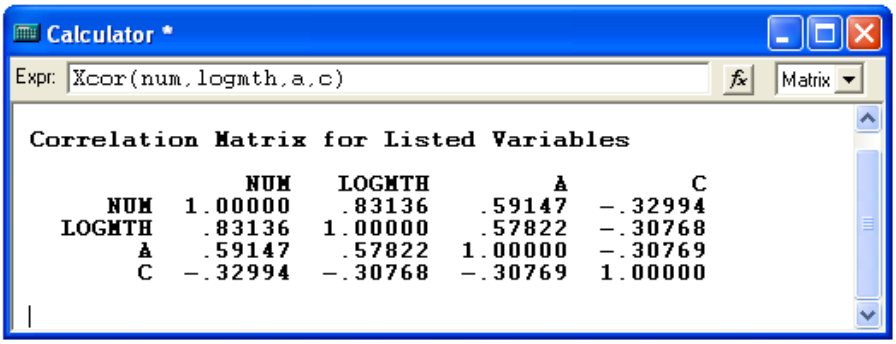


Figure 9.5 Matrix Result in the Calculator Window

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).

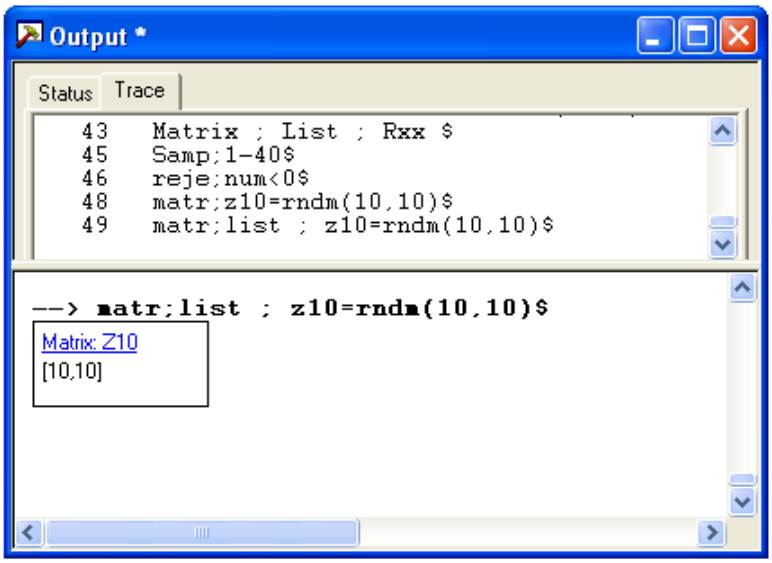


Figure 9.6 Matrix Result in the Output Window

If you double click the object, you can display the full matrix in a scaleable window.

9.2.4 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 R12.1. 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.

9.2.5 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 **POISSON** command from one of our earlier examples.

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 Poisson regression

```
+---------+--------------+----------------+--------+---------+----------+
|Variable | Coefficient  | Standard Error |b/St.Er.|P[|Z|>z] | Mean of X|
+---------+--------------+----------------+--------+---------+----------+
 Constant     -5.33586673      .69883851    -7.635   .0000
 C67            .74916894      .17622269     4.251   .0000     .29411765
 C72            .84733072      .18996097     4.461   .0000     .29411765
 C77            .36254858      .24697201     1.468   .1421     .14705882
 P77            .39897106      .12475671     3.198   .0014     .55882353
 LOGMTH         .84714538      .06903915    12.271   .0000    7.04925451
```

MATRIX ; Stat(b,varb) $

```
+-------------------------------------------------+
|Number of observations in current sample =      34 |
|Number of parameters computed here       =      10 |
|Number of degrees of freedom             =      24 |
+-------------------------------------------------+
```

```
+---------+-------------+----------------+--------+---------+
|Variable | Coefficient  | Standard Error |b/St.Er.|P[|Z|>z] |
+---------+-------------+----------------+--------+---------+
 B_1          -5.33586673      .69883851    -7.635   .0000
 B_6            .74916894      .17622269     4.251   .0000
 B_7            .84733072      .18996097     4.461   .0000
 B_8            .36254858      .24697201     1.468   .1421
 B_9            .39897106      .12475671     3.198   .0014
 B_10           .84714538      .06903915    12.271   .0000
```

MATRIX ; Stat(b,varb,x) $

```
+---------+-------------+----------------+--------+---------+
|Variable | Coefficient  | Standard Error |b/St.Er.|P[|Z|>z] |
+---------+-------------+----------------+--------+---------+
 Constant     -5.33586673      .69883851    -7.635   .0000
 C67            .74916894      .17622269     4.251   .0000
 C72            .84733072      .18996097     4.461   .0000
 C77            .36254858      .24697201     1.468   .1421
 P77            .39897106      .12475671     3.198   .0014
 LOGMTH         .84714538      .06903915    12.271   .0000
```

9.3 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

- **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×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×L where K and L are numbers of variables, or some other small values that you will define with your commands.*

9.3.1 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	GNP	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 *alldata* would be defined by the command

```
NAMELIST    ; demand = cons,invst,gnp
            ; alldata = year,cons,invst,gnp,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(10000×20 + 10000×10000), 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 $K \times K$.

9.3.2 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. 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 9.7, 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, $\mathbf{F} = (1/n)\mathbf{X'WX}$ 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.

9.4 Manipulating Matrices

The preceding described how to operate the matrix algebra package. The example in Section 9.1 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. In this section, we will list the various conventions that apply to these operations.

9.4.1 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.

Of course, 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.

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. (This is known as 'read only.')

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 $\mathbf{F} = (1/n)\mathbf{X}'\mathbf{X}$. 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.

9.4.2 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 $\mathbf{C} \times \mathbf{D}$ 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, $\mathbf{A}s\mathbf{B}'$, where *s* is a scalar, is the same as $s\mathbf{AB}'$. 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. Also, 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*
a'< *c* > *b*	=	*a' c*$^{-1}$ *b* *c* 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 9.1 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*.

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'c a 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

$$\begin{bmatrix}1 & 3\\ 3 & 5\end{bmatrix}\begin{bmatrix}2 & 4\\ 5 & 5\end{bmatrix} = \textbf{[1 / 3,5] * [2,4 / 5,5]}.$$

The multiplication operator sorts out scalars or 1 □1 matrices in a product. Consider, for example, **V** = **A(r′Ar)**$^{-1}$**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

$$\mathbf{b}_r = \mathbf{b}_u - (\mathbf{X'X})^{-1}\mathbf{R'}[\mathbf{R}(\mathbf{X'X})^{-1}\mathbf{R'}]^{-1}(\mathbf{Rb}_u - \mathbf{q}).$$

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 R12.1 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.

9.5 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*×1 column vector with all elements equal to value
*k*_value = a 1×*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 $a = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 5 \\ 4 & 5 & 6 \end{bmatrix}$.

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 $c1$ is a 5×2 matrix and $c2$ is 5×4. The matrix c=[$c1$,$c2$] 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

$$\mathbf{F} = \begin{bmatrix} \mathbf{C}_1 \\ \mathbf{C}_2 \end{bmatrix} \text{ use } \mathbf{f = [c1 / c2]}.$$

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].
```

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 R12.1 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.

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.

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*.

Diagonal Elements of a Matrix in a Vector

The function

a = Vecd (c)

creates column vector a from the diagonal elements of the matrix c.

Diagonal Matrix Created from a Vector

The command

a = 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.

9.6 Matrix Functions

There are roughly 50 functions that 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 9.1 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.

Characteristic Roots and Vectors

a = Cvec(c) - characteristic vectors.

If **C** is a $K \times K$ 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 $K \times 2$ 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 $[c_1, c_2, \ldots, c_Q]$ instead of a symmetric matrix, then A = Root(*c*) reports in a $K \times 2$ 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, $1 - c_1 z - c_2 z^2 - \ldots - c_Q z^Q = 0$, of the dynamic equation

$$y_t = c_1 y_{t-1} + c_2 y_{t-2} + \ldots + c_Q y_{t-Q} + \textit{other terms}.$$

The dominant root of the system is the largest reciprocal reported. If its modulus is larger than one, the equation is unstable.

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 = 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) $
```

9.7 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 $\mathbf{C'A^{-1}C}$ 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 9.2.

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 9.2 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.

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 $K\times1$ 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.

Xvcm(*x*) - covariance matrix for **X**
and Xcor(*x*) - correlation matrix for **X**

Chapter 10: Scientific Calculator

10.1 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 $\varepsilon \sim N[0,1]$, $\text{Prob}[y=1]=\Phi(\beta_0+\beta_1 x)$. 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 log L.

```
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.

10.2 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 10.1.

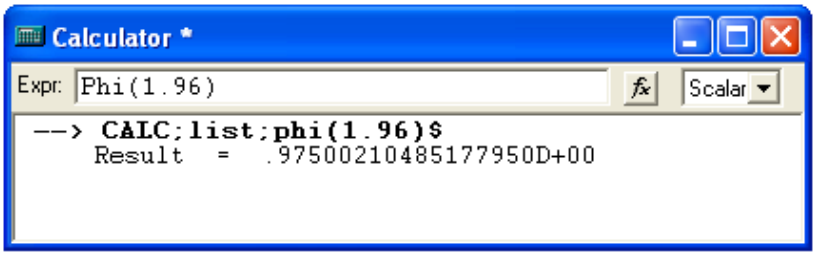


Figure 10.1 Calculator Window

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 10.2). Select Scalar to display the calculator functions. 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 10.2) 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 10.3. 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 R13.3 for details on the **; List**. Specification.

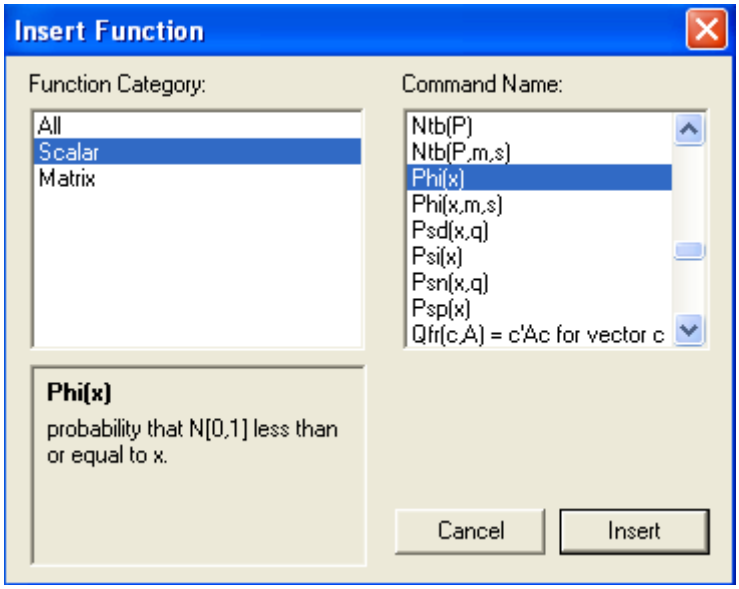


Figure 10.2 Insert Function Dialog Box for Calculator Functions

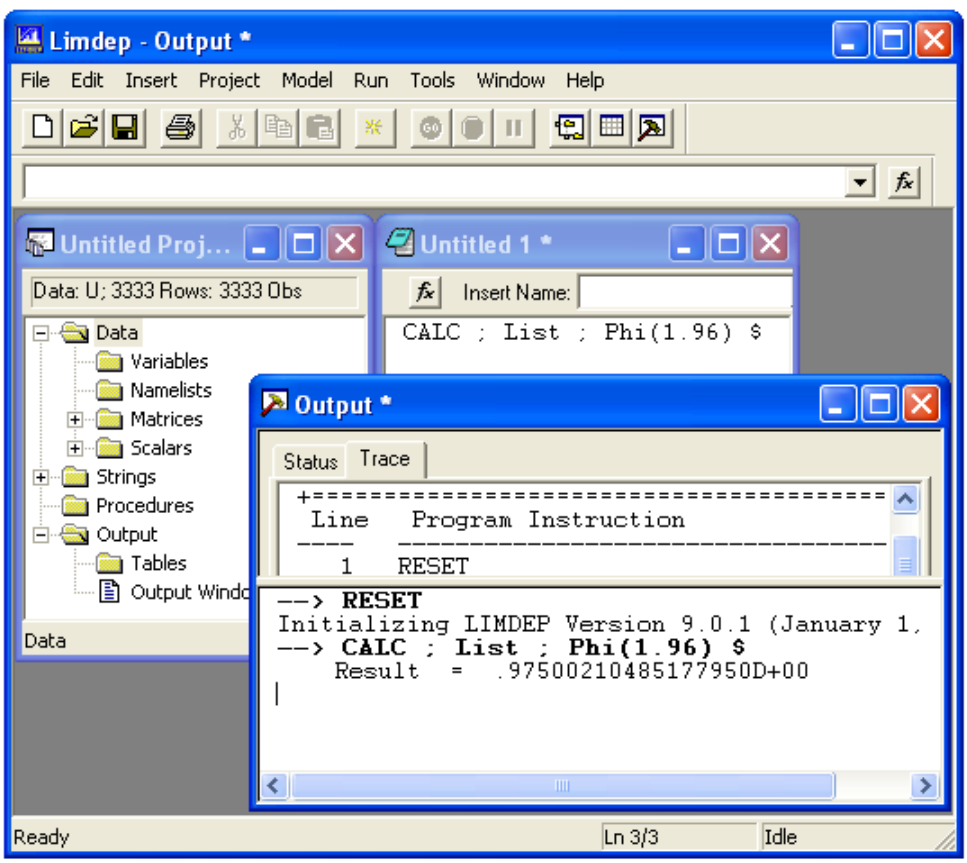


Figure 10.3 CALCULATE Command in Text Editor

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.)

10.3 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 R13.1. 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 Figure10.4.

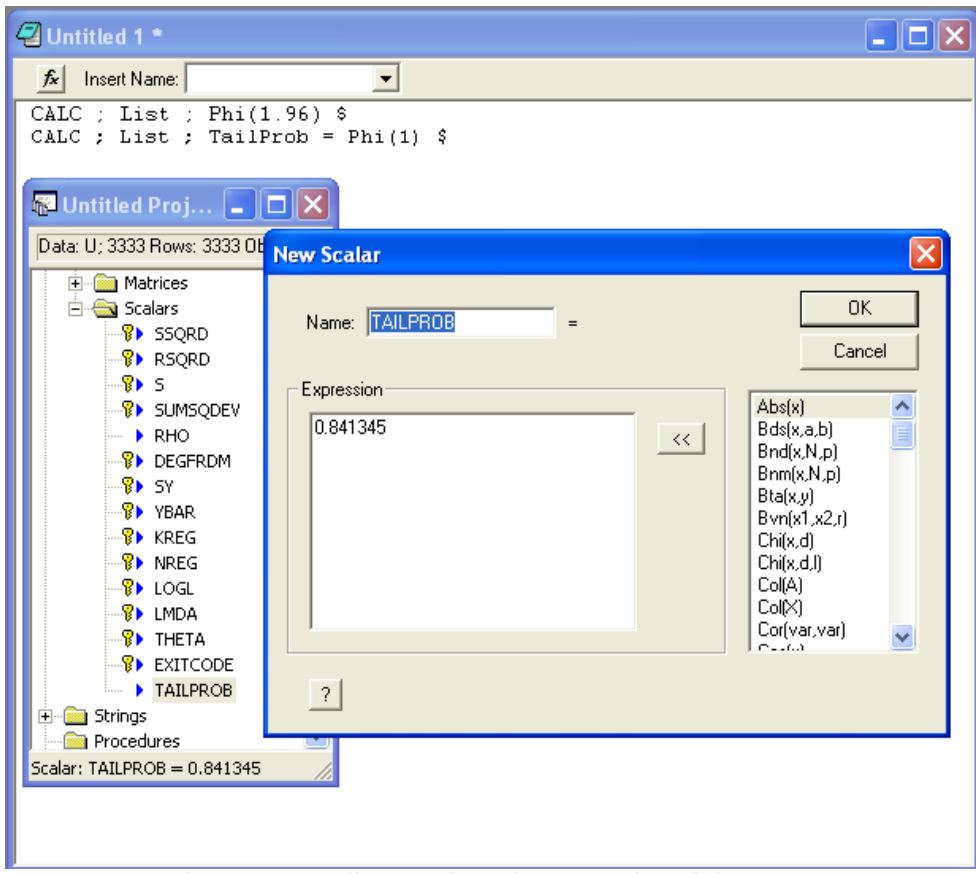


Figure 10.4 Edit Function of New Scalar Dialog Box

10.4 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 R5.2.2) and **REJECT** (see Section R7.4), 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 **CALCULATE** commands. This will be most useful in iterative programs to condition your **CALC** commands.

10.4.1 Reserved Names

You can have a total of 50 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 10.4. They are the ones marked as 'locked' with the gold key icon,. 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.

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*.

10.4.2 Work Space for the Calculator

Although there are 50 scalars available, the 14 protected names leave you a total of 36 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.

10.4.3 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 10.5 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.

10.5 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 9 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

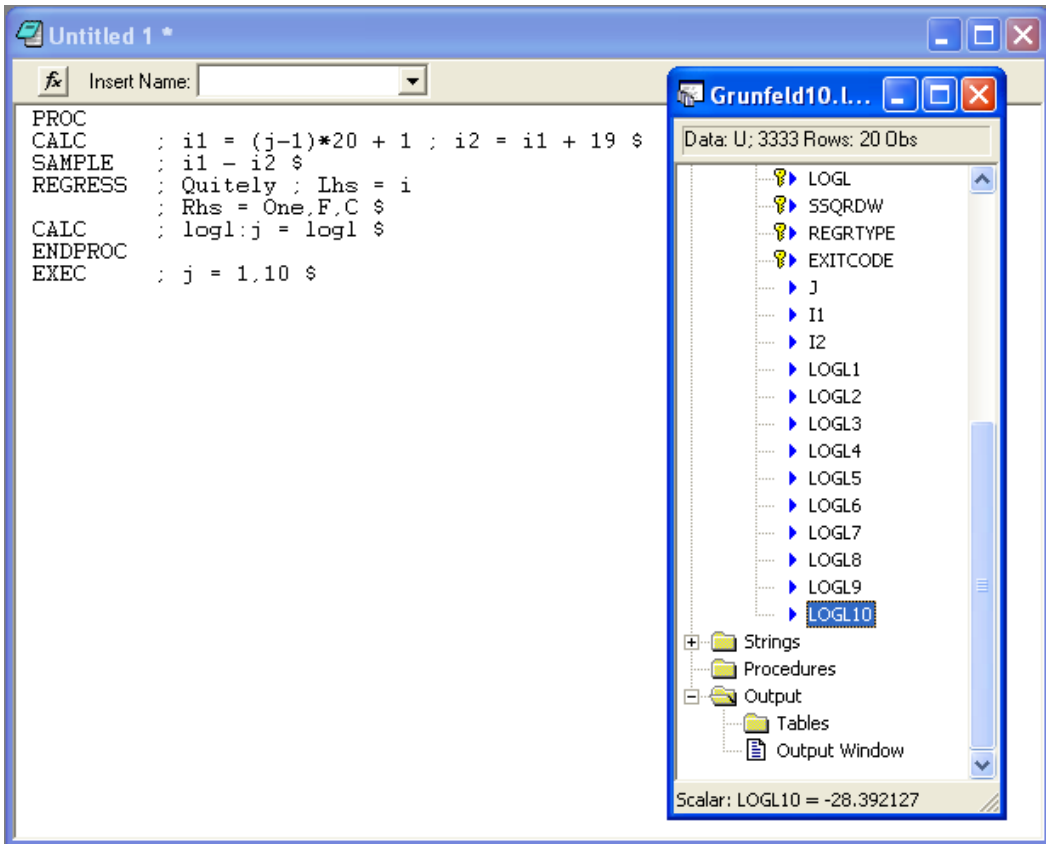


Figure 10.5 Procedure with Indexed Scalar Names

```
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 $\pi = 3.14159\ldots$ 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**.

10.6 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:

Basic Algebraic Functions

Log(x)	= natural log,	Exp(x)	= exponent,
Abs(x)	= absolute value,	Sqr(x)	= square root,
Sin(x)	= sine,	Cos(x)	= cosine,
Tan(x)	= tangent,	Rcs(x)	= arccosine,
Rsn(x)	= arcsine		
Ath(x)	= hyperbolic arctangent = ½ ln[(1+x)/(1-x)],		
Ati(x)	= inverse hyperbolic arctangent = [exp(2x)-1] / [exp(2x)+1].		

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 $x < 0$.

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,
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.

Probabilities and Densities for Continuous Distributions

Phi(x)	= probability that N[0,1] $\leq x$,
Phi(x,μ,σ)	= probability that N[μ,σ] $\leq x$,
N01(x)	= density of the standard normal evaluated at x (Note 'N-zero-one'),
Lgf(x)	= log of standard normal density = -½ (ln2π + 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 $\leq x$],
Chi(x,d)	= prob[chi squared variable with d degrees of freedom $\leq x$],
Fds(x,n,d)	= prob[F with n numerator and d denominator degrees of freedom $\leq x$],
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/θ $\leq x$],
Bds(x,α,β)	= prob[beta variable with parameters α,β $\leq x$].

Matrix Dimensions

If *A* is the name of a matrix,

Row(*A*)	= number of rows in matrix *A*,
Col(*A*)	= number of columns in matrix *A*.

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.

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(*x*1) and Xbr(*x*2) 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.

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[$1/n\Sigma_i \log(x_i)$],
Xhm(*variable*,*h*)	= the harmonic mean using parameter *h*; Xhm(x,h) = $[\Sigma_i x_i^{h}]^{1/h}$.

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.

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.

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-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.

10.7 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,

Chapter 11: Programming with Procedures

11.1 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 chapter 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.

11.2 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.

11.2.1 Placing Commands in the Editor

LIMDEP's editing window shown in Figure 11.1 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.

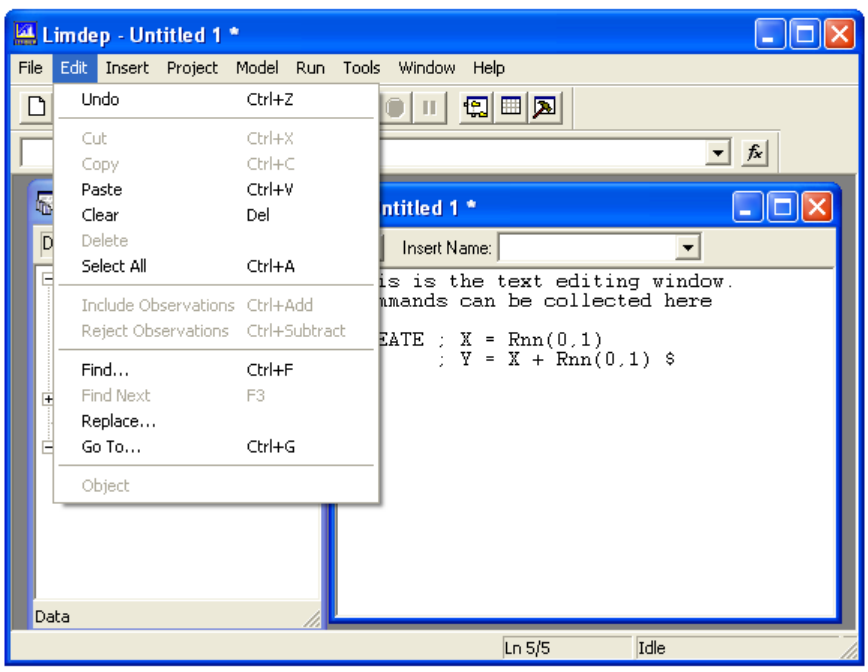


Figure 11.1 The Editing Window and the Edit Menu

The Insert menu shown in Figure 11.2 can also be used in the editing window. The Insert menu allows you to place specific items on the screen in the editing window:

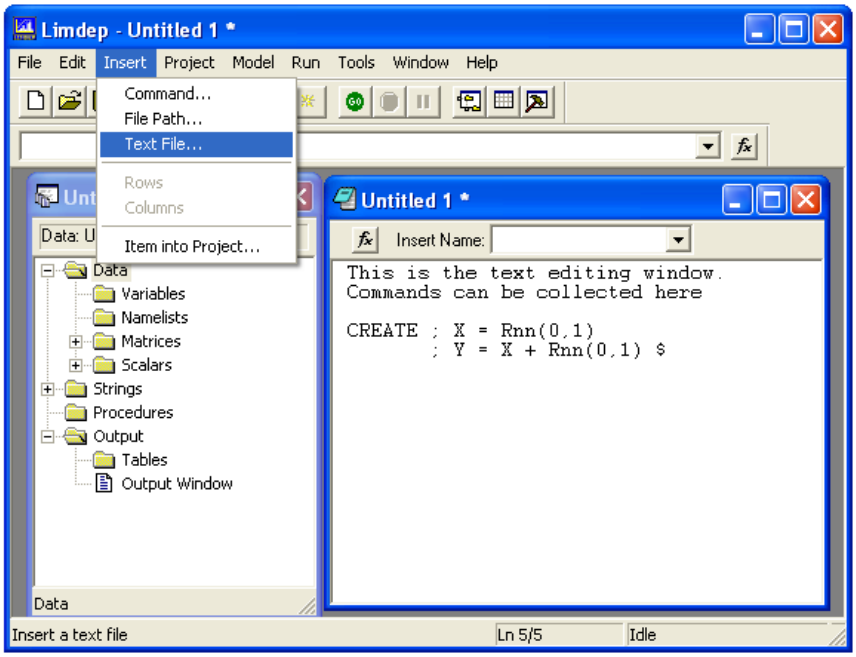


Figure 11.2 Insert Menu for Text Editor

- 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 verbs.
- 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 input files, or create input files, using this tool.

11.2.2 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 11.3.)
- Select the Run menu at the top of your screen. See Figure 11.4. 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.

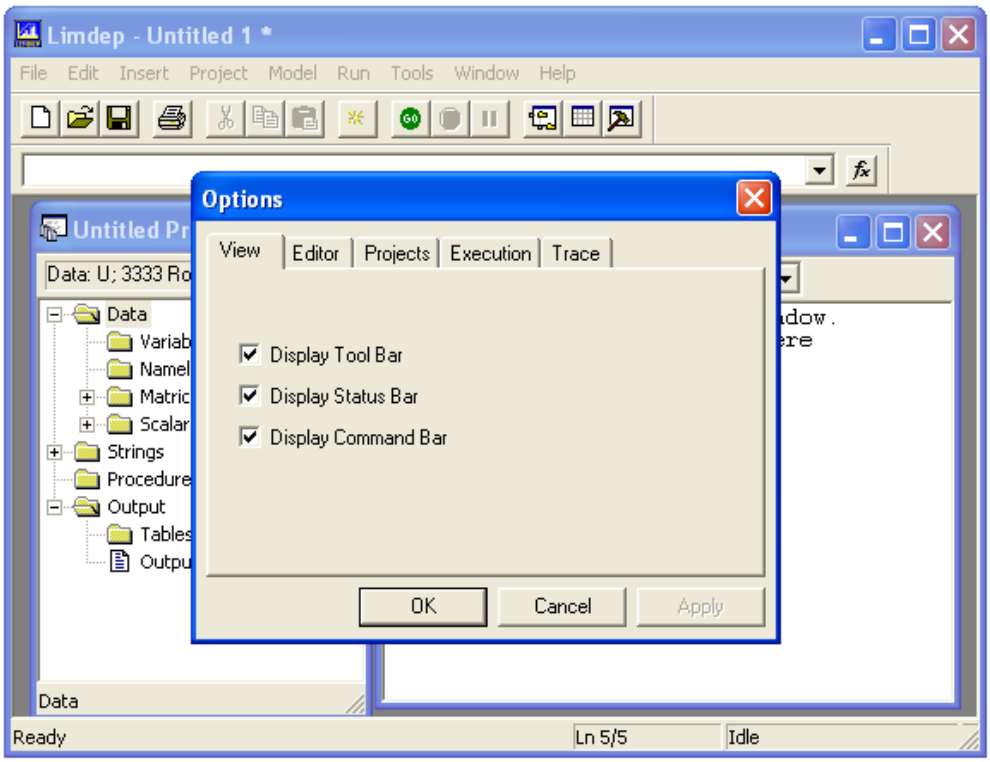


Figure 11.3 Tools:Options/View Menu to Set Up Desktop

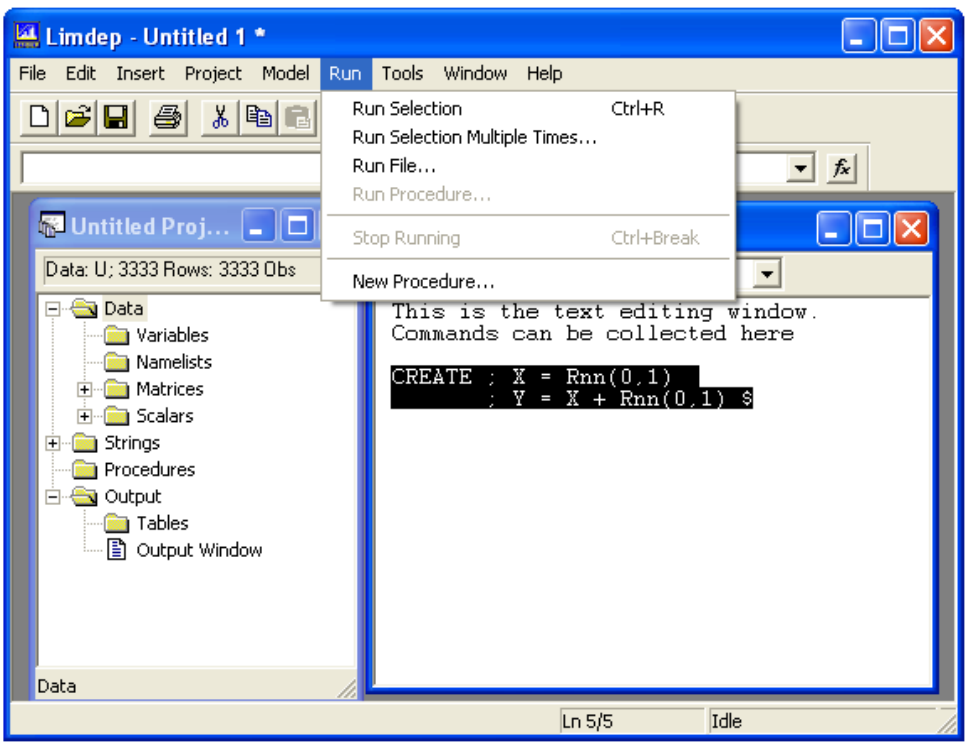


Figure 11.4 Run Menu

11.3 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. CALCULATE ; i = 0 $
Step 2. CALCULATE ; 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 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.

11.4 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.

Some notes on 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 2,500 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.

11.4.1 Executing a Procedure Silently

Procedures are often used to produce a final result with many intermediate computations. You can suppress intermediate output with

```
EXECUTE      ; Silent  $
```

This suppresses all output. When the procedure is completed, the **SILENT** switches are turned off. You can then use **MATRIX**, **CALC**, or whatever other means are necessary to inspect the desired final result from the procedure. You might use this in an experiment in which you fit the same model many (possibly thousands of) times and accumulate a statistic from the execution. For example, you might investigate whether the mean of a certain statistic is zero with the following procedure. The procedure is general – you could replace the application specific part with some particular estimation problem. It accumulates a result, then uses the central limit to test the hypothesis that the statistic being computed is drawn from a distribution with mean zero.

```
CALC        ; meanb = 0 ; sb = 0 ; nrep = 1000 $
PROC
  … generate the data set for the model command that computes the statistic
CALC        ; meanb = meanb + the statistic
            ; sb = sb + the statistic ^2 $
ENDPROC
EXECUTE     ; Silent ; n = nrep $
CALC        ; meanb = meanb / nrep
            ; sb = Sqr((sb – nrep * meanb^2)/(nrep – 1))
            ; List  ; z = Sqr(nrep) * meanb/sb  $
```

The procedure estimates the same model 1,000 times. The statistic of interest is *z*, computed at the last line. Since the model results are not useful, we use **; Silent** to suppress them. The number of repetitions is specified generically in a scalar named *nrep*, so if a larger or smaller sample is desired, it is necessary only to change the fixed value in the first line.

11.4.2 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) $
```

- 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) $
```

Chapter 12: Econometric Model Estimation

12.1 Introduction

The primary function carried out by *LIMDEP* is the estimation of econometric models. The first part of the documentation, the *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 there as well. This second part, the econometric modeling guide, will describe some specific modeling frameworks and instructions to be used for fitting these models.

The organization of this manual is by estimation framework, not by model command. We have found that users prefer that the program documentation be oriented toward the types of functions they want to perform, not to an alphabetical listing of commands. As such, you will find the arrangement of topics in this manual rather similar to the arrangement of topics in treatises in econometrics, such as Greene (2011). We begin with descriptive statistics in Chapter 13, various linear regression models in Chapters 14 and 15, and so on.

12.2 Econometric Models

This manual is devoted primarily to the methods by which you can use *LIMDEP* to fit equations to data and test hypotheses about the relationships implied by that estimation process. For purposes of documenting the program, we use the term 'model estimation' broadly, to encompass all those functions that involve manipulation of data to produce statistics to summarize the information the data contain. Thus, this manual begins with a chapter about computing descriptive statistics, which one might not normally consider model building. However, as data summaries, for program purposes, we consider these part of the model building functions in *LIMDEP*.

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. The remainder of this chapter will describe in general terms how to use this format to construct model estimation commands in *LIMDEP*.

12.3 Model Commands

LIMDEP's model commands all use the same format. The essential parts are as follows:

```
Model name     ; model variables specification
               ; essential specifications for some models
               ; optional specifications $
```

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.

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. To continue our example, a Poisson model might be specified using

```
POISSON      ; Lhs = patents ; Rhs = one , r_and_d $
```

which specifies one of the most well known applications of this model in economics. (The variable '*one*' is the constant term. We'll return to this below.) Some 'model' commands will have only one of these two specifications, such as

```
DSTAT        ; Rhs = patents $
```

which requests descriptive statistics for the variable *patents*. As can be seen here, we use the term 'model command' broadly to indicate analysis of a set of data, whether for description or parameter estimation. Other model commands might have only a Lhs variable, such as

```
SURVIVAL     ; Lhs = failtime $
```

which requests a nonparametric (life table) analysis of a variable named *failtime*. There are also many other types of variable specifications, such as

```
             ; Inst = a set of variable names
```

which will be used to specify the set of instrumental variables in the **2SLS** command.

Most models can be specified with nothing more than the model name and the identification of the essential variables. But, some models require additional specifications in order to be identified. For example, the specific model you want may be a particular case of a broad class of models and in order to specify it, you must provide the 'essential' specifications. For example, the basic command for survival modeling (with covariates to provide the 'model') would be

```
SURVIVAL     ; Lhs = failtime ; Rhs = one, usehours $
```

This form of the command is for Cox's proportional hazard model. In order to fit a parametric model, such as the Weibull model, you would use

```
SURVIVAL     ; Lhs = failtime ; Rhs = one, usehours
             ; Model = Weibull $
```

Note the last specification. This is the only way to specify a Weibull model, so for this model, this specification is essential. The Weibull model is requested as a type of survival model by this command. Obviously, not all models have mandatory specifications – the examples above do not. But, many do. The documentation in the chapters to follow will identify these.

Finally, all model frameworks have options which either extend the model itself or control how the model is estimated or how the estimation results are displayed. For example, the following fits a linear regression model and requests a robust estimator of the covariance matrix of the estimates:

```
REGRESS      ; Lhs = profit ; Rhs = one, sales ; Heteroscedasticity consistent  $
```

The latter specification does not change the model specification, it requests an additional computation, the White heteroscedasticity consistent estimator. For another example,

```
REGRESS      ; Lhs = profit ; Rhs = one, sales  ; Plot residuals $
```

fits a linear model and then plots the residuals. If the latter specification is omitted, the residuals will not be plotted.

12.4 Command Builders

In what follows, we will describe the model commands that you can use to fit the indicated model. In all cases, there are command builders that can be used instead of the text editor and command language. The command builders are described in Sections 4.4, 6.5.4 and 8.2.1.

12.5 Model Groups Supported by *LIMDEP*

A few of the various model commands and modeling frameworks supported by *LIMDEP* and *NLOGIT* are discussed in the chapters to follow. To suggest the breadth of the full menu, the following are the model names for the different classes of estimators supported by the program. Some, such as **HISTOGRAM** and **BURR** are quite narrow, single purpose instructions, while others, such as **POISSON**, call for large classes of models that may (as in this case) contain a large number of different variants. The specific models that are presented in Chapters 13 – 16 in this manual are highlighted in the list. The documentation below will present the basic forms of each of these. Additional specifications, options and model forms are developed in the full manual for *LIMDEP*.

Descriptive Statistics

CLASSIFY	Discriminant analysis – classification into latent groups.
DSTAT	Descriptive statistics.
TABLES	Descriptive statistics for stratified data.
CROSSTAB	Cross tabulations for discrete data.
HISTOGRAM	Histograms for discrete and continuous data.
KERNEL	Kernel density estimation of the density for a variable.
IDENTIFY	Descriptive statistics (ACF, PACF) for time series data.
SPECTRAL	Spectral analysis of a time series.

Plotting

FPLOT	Function plot for user specified function.
MPLOT	Scatter plot of matrices.
PLOT	Scatter or time plots of variables against each other.
SPLOT	Simultaneous scatter plots for several variables.

Linear Regressions and Variants

Single Equation

FRONTIER	Stochastic frontier models.
HREG	Heteroscedastic linear regression
QREG	Quantile regression.
REGRESS	Linear regression models (also **OLSQ** and **CRMODEL**).
TSCS	Time series/cross section, covariance structure models.
2SLS	Two stage (instrumental variable) estimation of linear models.

Multiple Equation

SURE	Linear seemingly unrelated regression models.
3SLS	Three stage (IV, GLS) estimator for systems of linear equations.

Sample Selection Models

MATCH	Propensity score matching to analyze treatment effects.
SELECT	Sample selection models with linear and tobit models.
INCIDENTAL	Incidental truncation (selection) model.
SWITCH	Switching regression models.

Nonlinear Regression, Optimization, Manipulation of Nonlinear Functions

ARMAX	Box Jenkins ARMA and dynamic linear equations.
BOXCOX	Regression based on the Box-Cox transformation of variables.
NLSQ	Nonlinear least squares for nonlinear regression models.
NLSURE	Nonlinear systems of equations, SURE or GMM estimation.

Analysis of Nonlinear Functions

FINTEGRATE	Function integration for user specified nonlinear function.
GMME	GMM estimation of model parameters.
MAXIMIZE	Maximization of user specified functions.
MINIMIZE	User defined minimization command.
WALD	Standard errors and Wald tests for user specified nonlinear functions.

Single Equation Models for Binary, Ordered and Multiple Discrete Choices

BINARY CHOICE	Simulation program for all binary choice estimators.
BIVARIATE	Bivariate probit models, partial observability models.
BURR	Burr model for binary choice.
CLOGIT	Multinomial logit model for discrete choice among multiple alternatives
COMPLOG	Complementary log log model for binary choice.
GOMPERTZ	Gompertz model for binary choice.
LOGIT	Binary and multinomial choice models based on the logistic distribution.
MLOGIT	Multinomial logit model.
MPROBIT	Multivariate probit model.
MSCORE	Maximum score semiparametric estimation for binary dependent variable.
NPREG	Nonparametric regression models.
ORDERED	Ordered probability models for ordered discrete choice.
PROBIT	Several forms of binary choice models.
SEMIPAR	Klein and Spady semiparametric estimator for binary choice.

Models for Count Data

GAMMA	Gamma model for count data.
NEGBIN	Negative binomial regression model.
POISSON	Models for count data.

Models for Censored Variables

BTOBIT	Bivariate tobit models.
GROUPED	Regression models for categorical censored data.
MIMIC	Multiple indicators and multiple causes for a latent variable.
NTOBIT	Nested tobit models.
TOBIT	Censored regression models.

Models for Variables with Limited Range of Variation

LOGLINEAR	Loglinear models, beta, gamma, Weibull, exponential, geometric, inverse Gaussian.
LOGNORMAL	Lognormal regression model.
TRUNCATE	Truncated regression models.

Models for Survival Times and Hazard Functions

SURVIVAL	Survival (hazard function) models.

12.6 Common Features of Most Models

There is wide variety in the components and features across the different model classes. But, there are several elements that are common to nearly all of them. Some of those are listed below.

12.6.1 Controlling Output from Model Commands

These two settings control model results that are generally not reported unless requested.

; Margin requests display of marginal effects.
; Printvc requests display of the estimated asymptotic covariance matrix. Normally, the matrix is not shown in the model results.

12.6.2 Robust Asymptotic Covariance Matrices

These settings control how the covariance matrix for the coefficient estimator is estimated..

; Cluster=spec requests computation of the cluster form of covariance the estimator.
; Str = spec is used with **; Cluster** to specify a stratified, two level form of data clustering.
; Robust requests a 'sandwich' estimator or robust covariance matrix for several discrete choice models.

12.6.3 Predictions and Residuals

Fitted values (predictions) and residuals from most single equation models are requested as follows:.

; List displays a list of fitted values with the model estimates.
; Keep = name keeps the fitted values as a new (or replacement) variable in the data set.
; Res = name keeps the residuals as a new (or replacement) variable.
; Prob = name saves the probabilities as a new (or replacement) variable for discrete choice models such as probit or logit.
; Fill requests that missing values or values outside the estimating sample be replaced by fitted values based on the estimated model.

Chapter 13: Describing Sample Data

13.1 Introduction

This chapter describes methods of obtaining descriptive statistics for one or more variables in your data set. Procedures are given for cross sections and for panel data.

13.2 Summary Statistics

The primary command for descriptive statistics is

DSTAT ; Rhs = list of variables $

This produces a table which lists for each variable, x_k, $k = 1,\ldots,K$, the basic statistics:

Sample mean $= \bar{x}_k = (1/N_k)\Sigma_{i=1}^{N_k} x_{ik}$,

Standard deviation $=, s_k = \sqrt{(1/(N-1)\Sigma_{i=1}^{N}(x_{ik} - \bar{x}_k)^2}$

Maximum value,

Minimum value,

Number of valid (nonmissing) cases.

Use

DSTAT ; Rhs = list of variables ; All $

to request, in addition, the skewness and kurtosis measures,

Sample skewness $= m_3 = \dfrac{\sum_{i=1}^{N_k}(x_{ik} - \bar{x}_k)^3/(N-1)}{s_k^3}$,

Sample kurtosis $= m_4 = \dfrac{\sum_{i=1}^{N_k}(x_{ik} - \bar{x}_k)^4/(N-1)}{s_k^4}$.

After the table of results is given, you may elect to display a covariance or correlation matrix (or both) for the variables. The request is added to the command:

; Output = 1 to obtain the covariance matrix,
; Output = 2 to obtain the correlation matrix,
; Output = 3 for both covariance and correlation matrices.

By adding

; All

to the command, we obtain the expanded results that include the skewness and kurtosis measures.

13.2.1 Weights

Weights may be used in computing all of the sums above by specifying

; Wts = name of weighting variable

Weights are always scaled so they sum to the current sample size. Thus, for example, the weighted mean would be

$$\bar{x}_k = (1/N_k)\Sigma_{i=1}^{N_k} w_{ik} x_{ik}$$

where $$w_{ik} = \frac{n_k z_{ik}}{\Sigma_{i=1}^{N_k} z_{ik}}$$

and z_{ik} = the weighting variable.

13.2.2 Missing Observations in Descriptive Statistics

In all cases, weighted or otherwise, sums are based on the valid observations. **DSTAT** automatically selects out the missing data. Most other models (save for those in *NLOGIT* 4.0 and most of the panel data estimators) do not routinely do so unless you have the **SKIP** switch set. Each variable may have a different number of valid cases, so the table of results gives the number for each one.

NOTE: The covariance and correlation matrices are based on the subset of observations for which there were no missing data for any variables. Each row in the table of results will list the number of valid cases used for that particular variable. Unfortunately, if different observations are missing for the various variables used in a covariance or correlation matrix, the union of the observations for which all variables are present can contain very few observations. For better or worse, this union is the set of observations used in computing the matrices.

If your data contain missing values, the scaling in the previous section is automatically adjusted for each variable. Moments are scaled by the number of valid observations or sum of weights for that variable.

13.2.3 Sample Quantiles

You may obtain more detailed statistics about variables by requesting the sample quantiles. This feature produces sample order statistics and the deciles and quartiles of the sample of values for each variable. The keyword in the command is

; Quantiles

For example:

```
CALC      ; Ran(12345) $
SAMPLE    ; 1 – 1000 $
CREATE    ; x = Rnn(0,1) $
CREATE    ; y = Rnn(0,1) ^ 2 $
DSTAT     ; Rhs  = x, y ; Quantiles  ; Matrix $
```

The **CREATE** commands produce random samples from the standard normal and the chi squared[1] distributions. The following output results:

```
Descriptive Statistics
All results based on nonmissing observations.
================================================================================
Variable        Mean         Std.Dev.        Minimum         Maximum      Cases
================================================================================
All observations in current sample
--------------------------------------------------------------------------------
X       | -.177875E-01  .972540     -3.14333      3.02263         1000       0
Y       |  .983443      1.42493      .162304E-07  11.6616         1000       0
```

Matrix: LastDsta
[2,7]

```
Order Statistics for Variables
Percentile     X            Y
Min.          -3.1433       .16230E-07
10th          -1.2680       .11610E-01
20th          -.84366       .51309E-01
25th          -.68126       .83422E-01
30th          -.53522       .13160
40th          -.26384       .28056
Med.           -.48192E-01   .43805
60th           .25616       .71992
70th           .51574       1.0797
75th           .65565       1.3230
80th           .83508       1.6071
90th          1.1866        2.5999
Max.           3.0226        11.662
Partition of range Minimum to Maximum
Range of X     X            Y
Minimum       -3.1433       .16230E-07
1st.Qrtl      -1.6018       2.9154
Midpoint       -.60348E-01   5.8308
3rd.Qrtl       1.4811       8.7462
Maximum        3.0226        11.662
```

Matrix: LastQntl
[13,2]

13.3 Histograms

The command for computing and plotting a histogram for a variable is

HISTOGRAM ; Rhs = the variable $

LIMDEP computes two types of histograms, for discrete (count) and for continuous data. The default type is for a frequency count of discrete data. Data are assumed to be coded 0,1,…,99. Values less than zero or above 99 are treated as out of range. A count of invalid observations is given with the output of the command. Continuous variables are assigned to 40 equal width intervals over the range of the variables. The histogram is accompanied by a table listing the relative frequencies and cumulated frequencies.

To illustrate the use of this feature, we use a data set that was employed in the study of health care system utilization, Regina T. Riphahn, Achim Wambach, and Andreas Million, 'Incentive Effects in the Demand for Health Care: A Bivariate Panel Count Data Estimation,' *Journal of Applied Econometrics*, Vol. 18, No. 4, 2003, pp. 387-405. The raw data were downloaded from the journal's data archive website:

http://qed.econ.queensu.ca/jae/2003-v18.4/riphahn-wambach-million

The data, which will be used in several applications below, are an unbalanced panel of observations on health care utilization by 7,293 individuals. The group sizes in the panel number as follows: T_i: 1=1525, 2=2158, 3=825, 4=926, 5=1051, 6=1000, 7=987. There are altogether 27,326 observations. Variables in the file are

hhninc	=	household nominal monthly net income in German marks / 10000.
Hhkids	=	children under age 16 in the household = 1, otherwise = 0,
educ	=	years of schooling
married	=	marital status
female	=	1 for female, 0 for male
docvis	=	number of visits to the doctor
doctor	=	number of doctor visits > 0
hospvis	=	number of visits to the hospital

13.3.1 Histograms for Continuous Data

A histogram for the continuous variable *hhninc* would appear as follows:

HISTOGRAM ; Rhs = hhninc $

You can select the number of bars to plot with

; Int = k

This can produce less than satisfactory results, however.

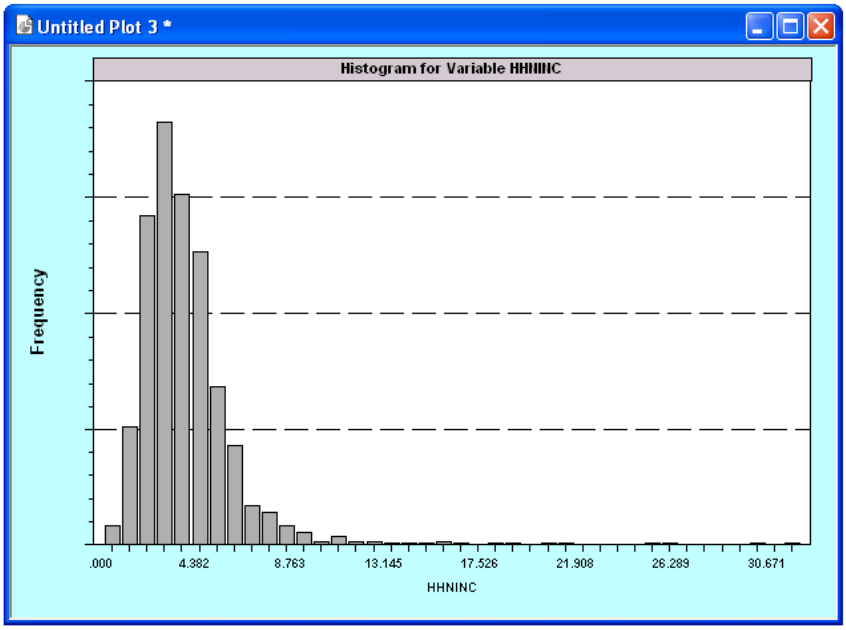


Figure 13.1 Histogram for a Continuous Variable

There are other ways to examine continuous data. One way is to use **RECODE** to change your continuous variable into a discrete one. Alternatively, you may provide a set of interval limits and request a count of the observations in the intervals you define. The command would be

HISTOGRAM ; Rhs = … ; Limits = l0,l1,…,lk $

where the limits you give are the left boundaries of the intervals. Thus, the number of limits you provide gives the number of intervals. Intervals are defined as 'greater than or equal to lower and less than upper.' For example, still using our income data,

HISTOGRAM ; Rhs = educ ; Limits = 0, .75, 2.5, 4, 6, 9, 12 $

defines five bins for the histogram, with the rightmost containing all values greater than or equal to 12. To request *K* equal length intervals in the range *lower* to *upper*, use

HISTOGRAM ; Rhs = variable ; Int = k ; Limits = lower,upper $

Finally, you can use **HISTOGRAM** to search for the interval limits instead of the frequency counts. The command

HISTOGRAM ; Rhs = variable ; Bin = p $

where '*p*' is a sample fraction (proportion), will obtain the interval boundaries such that each bin contains the specified proportion of the observations.

NOTE: If the specified proportion does not lead to an even set of bins, then an extra, smaller bin is created if the remaining proportion is more than $p/2$. For example, if p is .22, there will be four bins with .22 and one at the right end with .12. But, if the extra mass is less than $p/2$, it is simply added to the rightmost bin, as for p = .16, for which the sixth bin will contain .2 of the observations.

13.3.2 Histograms for Discrete Data

The data are also inspected to determine the type and the correct number of bars to plot for a discrete variable. For a discrete variable, the plot can be exact. Once again, up to 99 bars may be displayed: For example, the count of doctor visits in the health care data appear as follows:

HISTOGRAM ; Rhs = docvis $

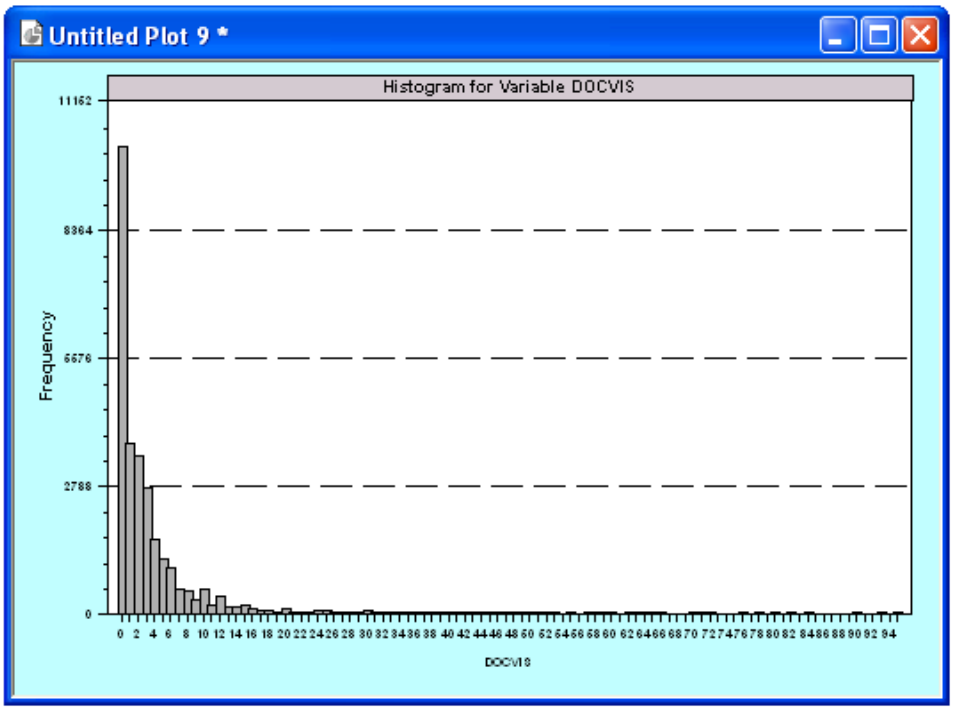


Figure 13.2 Histogram for a Discrete Variable

The long tail of the skewed distribution has rather distorted the figure. The options described earlier can be used to modify the figure. However, those options are assumed to be used for continuous data, which would distort the figure in another way. A more straightforward way to deal with the preceding situation is to operate on the data directly. For example,

HISTOGRAM ; If [docvis <= 25] ; Rhs = docvis $

truncates the distribution, but produces a more satisfactory picture of the frequency count.

13.4 Cross Tabulations

The command for crosstabs based on two variables is

CROSSTAB ; Lhs = rows variable ; Rhs = columns variable $

Use **CROSSTAB** to analyze a pair discrete of variables that are coded 0,1,… up to 49 (i.e., up to 2,500 possible outcomes). The table may be anywhere from 2×2 to 50×50. (Row and column sizes need not be the same.) Observations which do not take these values are tabulated as 'out of range.'

This command assumes that your data are coded as integers, 0,1,… If you wish to analyze continuous variables, you must use the **RECODE** command to recode the continuous ranges to these values.

The categories are automatically labeled 'NAME=0,' 'NAME=1,'…, etc. for the two variables. To provide your own labels and to specify the number of categories for the variables, add

; Labels = list of labels for Lhs / list of labels for Rhs

to the command. Labels may contain up to eight characters. Separate labels in the lists with commas. Cross tabulations may be computed with unequally weighted observations. The specification is

; Wts = variable

as usual. This scales the weights to sum to the sample size. If the weights are replications that should not be scaled, use

; Wts = variable,Noscale

13.5 Kernel Density Estimation

The command for kernel density estimation

KERNEL ; Rhs = the variable $

The kernel density estimator is a device used to describe the distribution of a variable nonparametrically, that is, without any assumption of the underlying distribution. The kernel density function for a single variable is computed using

$$f(z_j) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left[\frac{(z_j - x_i)}{h}\right], j = 1,\ldots,M.$$

The function is computed for a specified set of values z_j, j = 1,…,*M*. Note that each value requires a sum over the full sample of *n* values. *The default value of M is 100*. The primary component of the computation is the kernel function, *K*[.], a weighting function that integrates to one.

Eight alternatives are provided:

1. Epanechnikov:	$K[z]$	$= .75(1 - .2z^2) / \sqrt{5}$ if $\|z\| \leq 5$, 0 else,
2. Normal:	$K[z]$	$= \phi(z)$ (normal density), $-\infty < z < \infty$
3. Logit:	$K[z]$	$= \Lambda(z)[1-\Lambda(z)]$ (default) , $-\infty < z < \infty$
4. Uniform:	$K[z]$	$= .5$ if $\|z\| \leq 1$, 0 1 else,
5. Beta:	$Z[z]$	$= (1-z)(1+z)/24$ if $\|z\| < 1$, 0 1 else,
6. Cosine:	$K[z]$	$= 1 + \cos(2\pi z)$ if $\|z\| < .5$, 0 else,
7. Triangle:	$K[z]$	$= 1 - \|z\|$, if $\|z\| \leq 1$, 0 else.
8. Parzen:	$K[z]$	$= 4/3 - 8z^2 + 8\|z\|^3$ if $\|z\| \leq .5$, $8(1-\|z\|)^3$ else.

The other essential part of the computation is the smoothing (bandwidth) parameter, *h*. Large values of *h* stabilize the function, but tend to flatten it and reduce the resolution (in the same manner as its discrete analog, the bin width in a histogram). Small values of *h* produce greater detail, but also cause the estimator to become less stable.

The basic command is

KERNEL ; Rhs = the variable $

With no other options specified, the routine uses the logit kernel function, and uses a data driven bandwidth equal to

$$h = .9Q/n^{0.2} \text{ where } Q = \min(std.dev., range/1.5)$$

For an example, we will compute the kernel density that is a smoothed counterpart to the histogram for income distribution in Figure 13.1. The command is

KERNEL ; Rhs = hhninc $

The results follow. The histogram is repeated to show the similarity. Once again, the very long tail of the distribution distorts the figure. We show below how to adjust the parameters to accommodate this problem. For the figure below, we used

; Endpoints = 0,30

to force the figures to have the same range of variation on the horizontal axis.

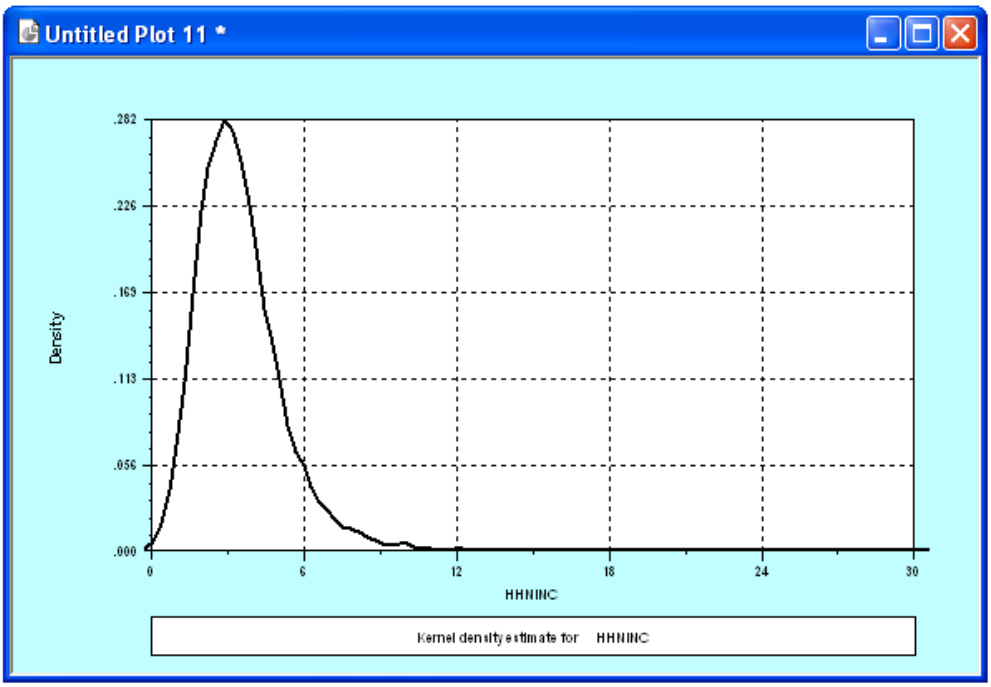
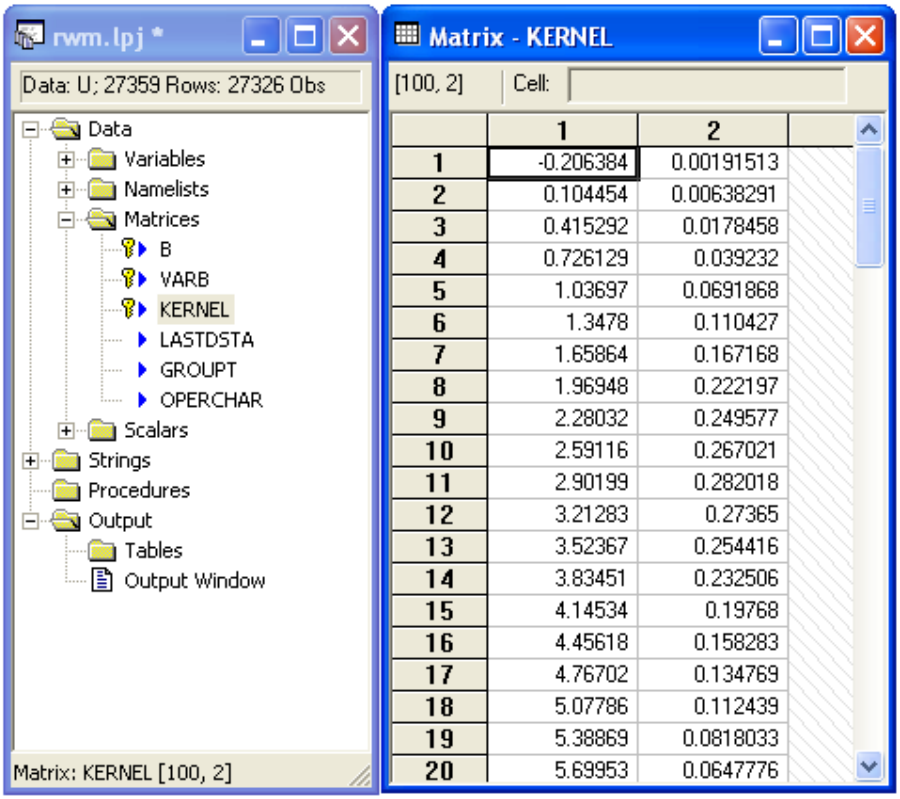


Figure 13.3 Kernel Density Estimator for Incomes

The kernel density also produces some summary statistics, as shown below for the example in Figure in 13.3.

```
+--------------------------------------+
| Kernel Density Estimator for HHNINC  |
| Observations       =         27326   |
| Points plotted     =           100   |
| Bandwidth          =       .206384   |
| Statistics for abscissa values----   |
| Mean               =      3.520836   |
| Standard Deviation =      1.769083   |
| Minimum            =       .000000   |
| Maximum            =     30.671000   |
| ---------------------------------    |
| Kernel Function    =      Logistic   |
| Cross val. M.S.E.  =       .000000   |
| Results matrix     =        KERNEL   |
+--------------------------------------+
```

The data used to plot the kernel estimator are also retained in a new matrix named (of course) *kernel*. Figure 13.4 shows the results for the preceding plot:



	1	2
1	-0.206384	0.00191513
2	0.104454	0.00638291
3	0.415292	0.0178458
4	0.726129	0.039232
5	1.03697	0.0691868
6	1.3478	0.110427
7	1.65864	0.167168
8	1.96948	0.222197
9	2.28032	0.249577
10	2.59116	0.267021
11	2.90199	0.282018
12	3.21283	0.27365
13	3.52367	0.254416
14	3.83451	0.232506
15	4.14534	0.19768
16	4.45618	0.158283
17	4.76702	0.134769
18	5.07786	0.112439
19	5.38869	0.0818033
20	5.69953	0.0647776

Figure 13.4 Matrix Results from KERNEL

You may specify the kernel function to be used with

; Kernel = one of the eight types of kernels listed earlier

The bandwidth may be specified with

; Smooth = the bandwidth parameter

The default number of points specified is 100, with *zj* = a partition of the range of the variable
You may specify the number of points, up to 200 with

; Pts = number of points to compute and plot

13.6 Scatter Plots and Plotting Data

This chapter will describe commands for producing high resolution graphs. This feature can be used for simple scatter diagrams, time series plots, and for plotting functions such as log likelihoods. You can print the graphics on standard printers and create plotter files for export to word processing programs such as Microsoft *Word*.

13.6.1 Printing and Exporting Figures

LIMDEP uses the standard Windows interface between input and output devices. When a plot appears in a window, you can use File:Print to send a copy to your printer. You can also save the graph as a Windows metafile (.WMF format) by using File:Save or File:Save As. For purposes of illustrating these functions, we will use Figure 13.5 which was generated by a **PLOT** command. The figure shows *LIMDEP*'s base format for graphics. Every graph generated is placed in its own scalable window, as shown in Figure 13.6, apart from the project, editing and output windows already open. This window will remain open until you close it. When you are finished reviewing the figure, you should close the window to avoid proliferating windows. You will be prompted to save the graph if you have not already done so.

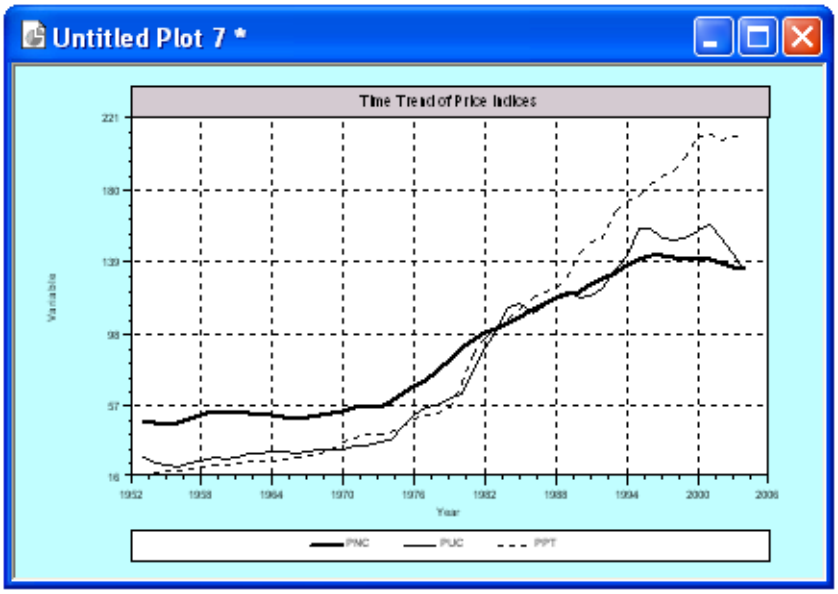


Figure 13.5 Time Series Plot Using the PLOT Command

13.6.2 Saving a Graph as a Graphics File

You may save any figure from a graphics window to disk in the Windows metafile (.wmf) format. Use File:Save or File:Save As. This file type is transportable to many other programs, including Microsoft *Word* and *Excel*. For example, in *Word*, click the Insert menu and then select Picture, then From File to import your .wmf file into your *Word* document. The Windows .wmf format includes codes that allow you to scale the figure to whatever size you desire.

TIP: Using Edit/Copy in *LIMDEP* then Edit/Paste in *Word* or *Excel* will transport the graph from the screen to your document. However, the internal components of the figure are not fully formed by this method, and the quality of the figure will be inferior to what you will obtain by writing the figure to a .wmf file and importing the file into the other program.

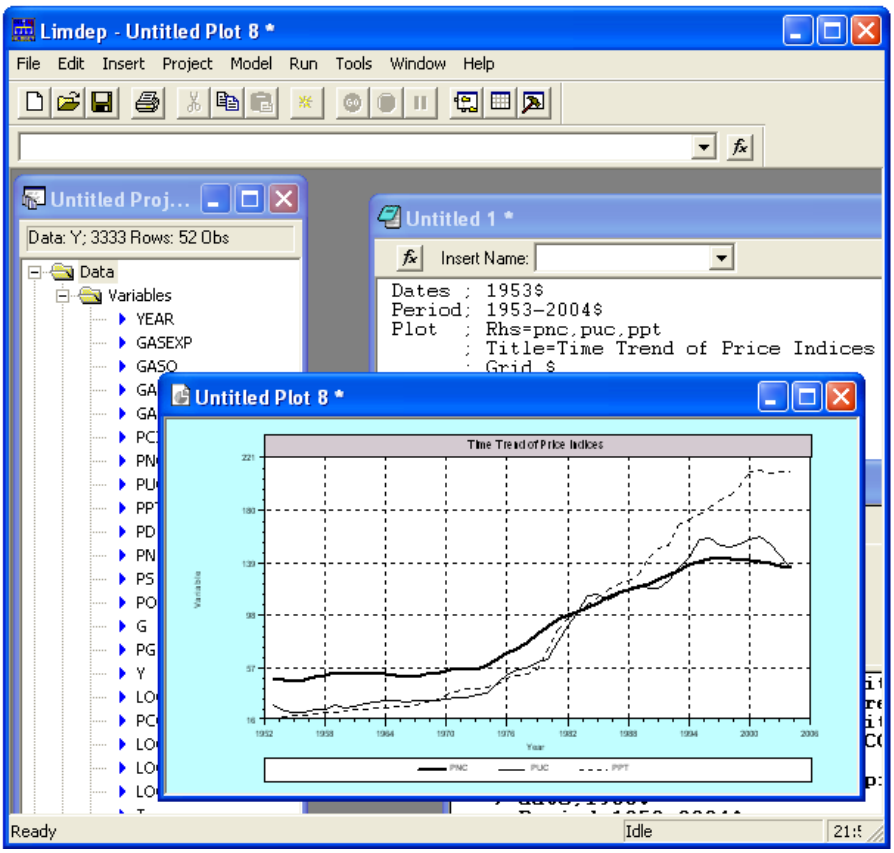
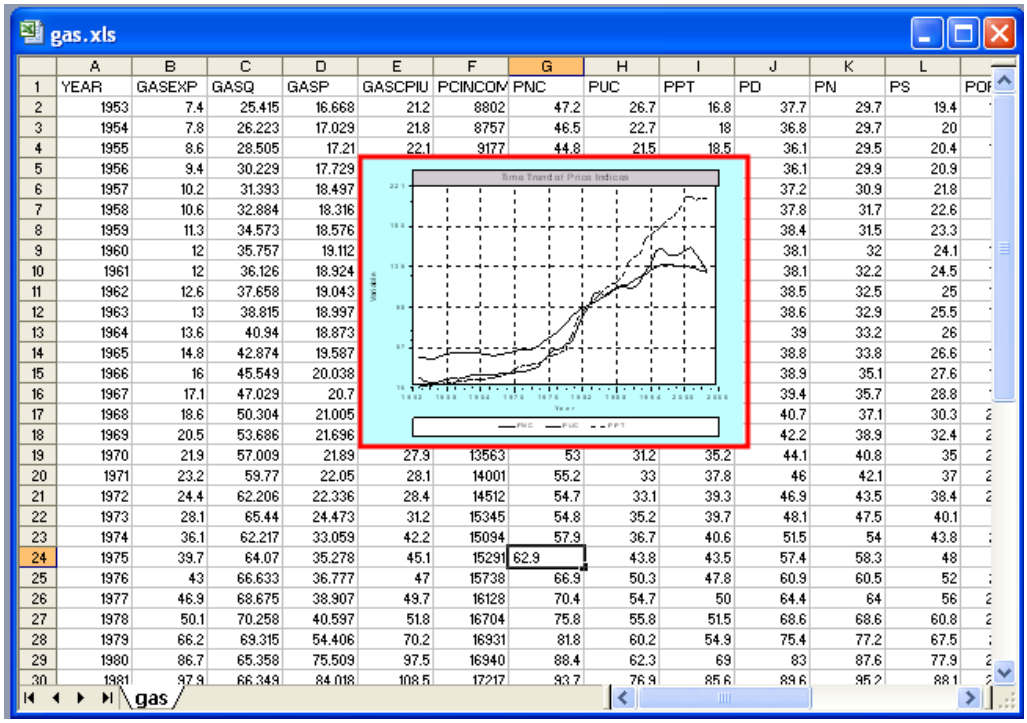


Figure 13.6 *LIMDEP* Desktop with Graphics Window



	A	B	C	D	E	F	G	H	I	J	K	L	
1	YEAR	GASEXP	GASQ	GASP	GASCPIU	PCINCOM	PNC	PUC	PPT	PD	PN	PS	POP
2	1953	7.4	25.415	16.668	21.2	8802	47.2	26.7	16.8	37.7	29.7	19.4	
3	1954	7.8	26.223	17.029	21.8	8757	46.5	22.7	18	36.8	29.7	20	
4	1955	8.6	28.505	17.21	22.1	9177	44.8	21.5	18.5	36.1	29.5	20.4	
5	1956	9.4	30.229	17.729						36.1	29.9	20.9	
6	1957	10.2	31.393	18.497						37.2	30.9	21.8	
7	1958	10.6	32.884	18.316						37.8	31.7	22.6	
8	1959	11.3	34.573	18.576						38.4	31.5	23.3	
9	1960	12	35.757	19.112						38.1	32	24.1	
10	1961	12	36.126	18.924						38.1	32.2	24.5	
11	1962	12.6	37.658	19.043						38.5	32.5	25	
12	1963	13	38.815	18.997						38.6	32.9	25.5	
13	1964	13.6	40.94	18.873						39	33.2	26	
14	1965	14.8	42.874	19.587						38.8	33.8	26.6	
15	1966	16	45.549	20.038						38.9	35.1	27.6	
16	1967	17.1	47.029	20.7						39.4	35.7	28.8	
17	1968	18.6	50.304	21.005						40.7	37.1	30.3	
18	1969	20.5	53.686	21.696						42.2	38.9	32.4	
19	1970	21.9	57.009	21.89	27.9	13563	53	31.2	35.2	44.1	40.8	35	
20	1971	23.2	59.77	22.05	28.1	14001	55.2	33	37.8	46	42.1	37	
21	1972	24.4	62.206	22.336	28.4	14512	54.7	33.1	39.3	46.9	43.5	38.4	
22	1973	28.1	65.44	24.473	31.2	15345	54.8	35.2	39.7	48.1	47.5	40.1	
23	1974	36.1	62.217	33.059	42.2	15094	57.9	36.7	40.6	51.5	54	43.8	
24	1975	39.7	64.07	35.278	45.1	15291	62.9	43.8	43.5	57.4	58.3	48	
25	1976	43	66.633	36.777	47	15738	66.9	50.3	47.8	60.9	60.5	52	
26	1977	46.9	68.675	38.907	49.7	16128	70.4	54.7	50	64.4	64	56	
27	1978	50.1	70.258	40.597	51.8	16704	75.8	55.8	51.5	68.6	68.6	60.8	
28	1979	66.2	69.315	54.406	70.2	16931	81.8	60.2	54.9	75.4	77.2	67.5	
29	1980	86.7	65.358	75.509	97.5	16940	88.4	62.3	69	83	87.6	77.9	
30	1981	97.9	66.349	84.018	108.5	17217	93.7	76.9	85.6	89.6	95.2	88.1	

Figure 13.7 *Excel* Spreadsheet with *LIMDEP* .mf File Imported

13.6.3 The PLOT Command

The command for producing a basic scatter (*XY*) plot of one or more variables against another variable is

PLOT ; Lhs = variable on horizontal axis
; Rhs = variables (up to five) on vertical axis $

Note the reversal of LIMDEP's usual convention. This command puts the Lhs variable on the horizontal axis, whereas a regression might be expected to do the reverse.

You may add a title to the figure by including

; Title = the title to be used

The title is placed at the top of the figure. The vertical axis of the plot is usually labeled with some variable name. You can override this with

; Yaxis = the label to be used, up to eight characters

This will often be useful when you plot a function or more than one variable.

13.6.4 Plotting One Variable Against Another

To produce a scatter plot of one variable (*y*) against another (*x*) variable, the **PLOT** command is given with only a single Rhs variable. The command would be

```
PLOT      ; Lhs = x  ; Rhs  =  y $
```

Figure E13.5 was produced with the commands listed.

```
CREATE    ; g     = gasq/(100*pop/282429)$
PLOT      ; Lhs   = gasp ; Rhs = g
          ; Title = Simple Plot of Gas against Price $
```

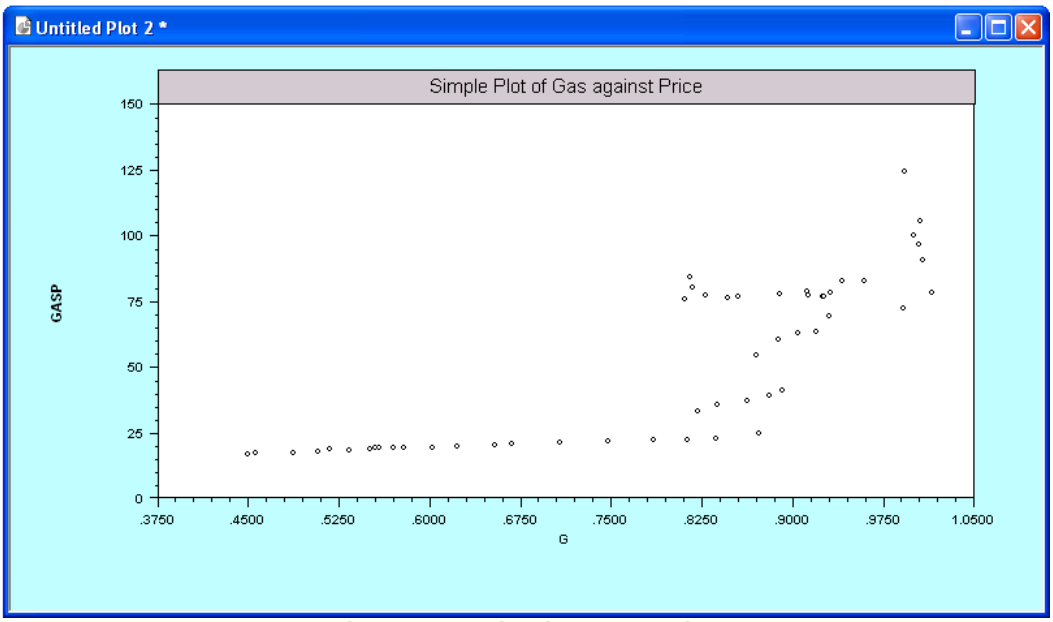


Figure 13.8 Simple Scatter Plot

13.6.5 Plotting a Simple Linear Regression

To add a regression line to a figure, add

```
; Regression
```

to the **PLOT** command. By adding **; Regression** to the preceding command, we obtain the plot in Figure 13.9. (You can also obtain this by selecting Display linear regression line in the Options page of the command builder.) (In previous versions of *LIMDEP*, the regression equation would replace the title in the figure. In this version, the title appears as a header and the regression equation is placed in the footer of the figure.)

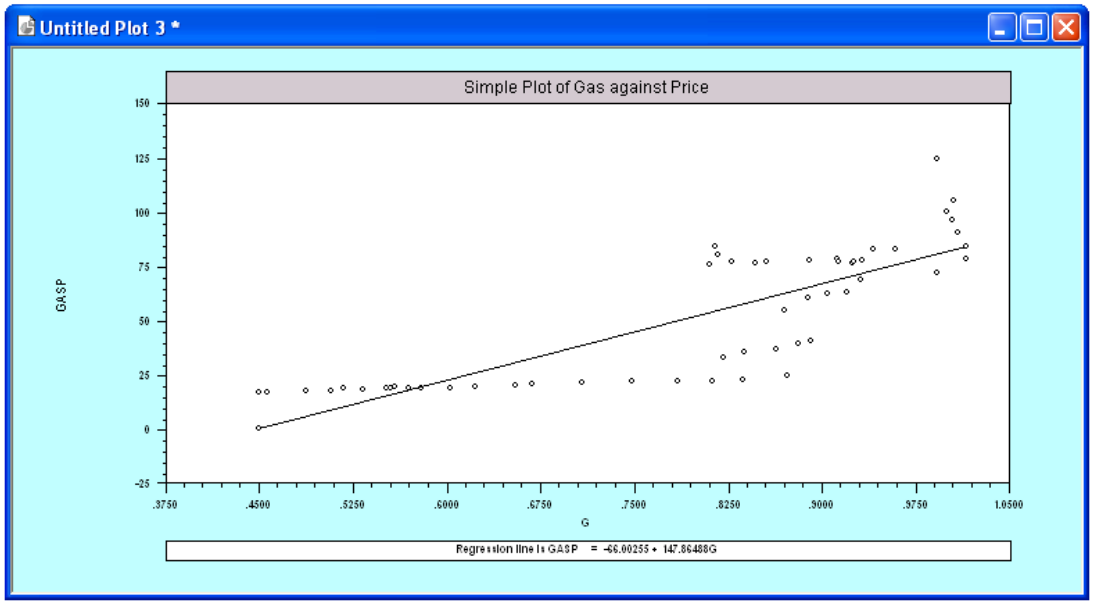


Figure 13.9 Scatter Plot with Linear Regression

13.6.6 Time Series Plots

Time series plots, that is, plots of variables against the date can be obtained by using **DATES** and **PERIOD** to set up the dating, then omitting the **; Lhs** part of the **PLOT** command. When you omit the **; Lhs** part of the command, it is assumed that this is a time series plot, and the adjacent points are automatically connected. The figure is also automatically labeled with the dates. The **; Fill** specification discussed below is not necessary. Figure 13.10 is a time series plot of the three macroeconomic price series for the data above. Note the use of the **; Grid** specification to improve the readability of the figure. (See Section E4.3.5 for details on this specification.) The command is

```
DATES     ; 1953 $
PERIOD    ; 1953 – 2004 $
PLOT      ; Rhs   = pn, pd, ps
          ; Grid
          ; Title  = Time Series Plot of Price Series
          ; Yaxis = Prices $
```

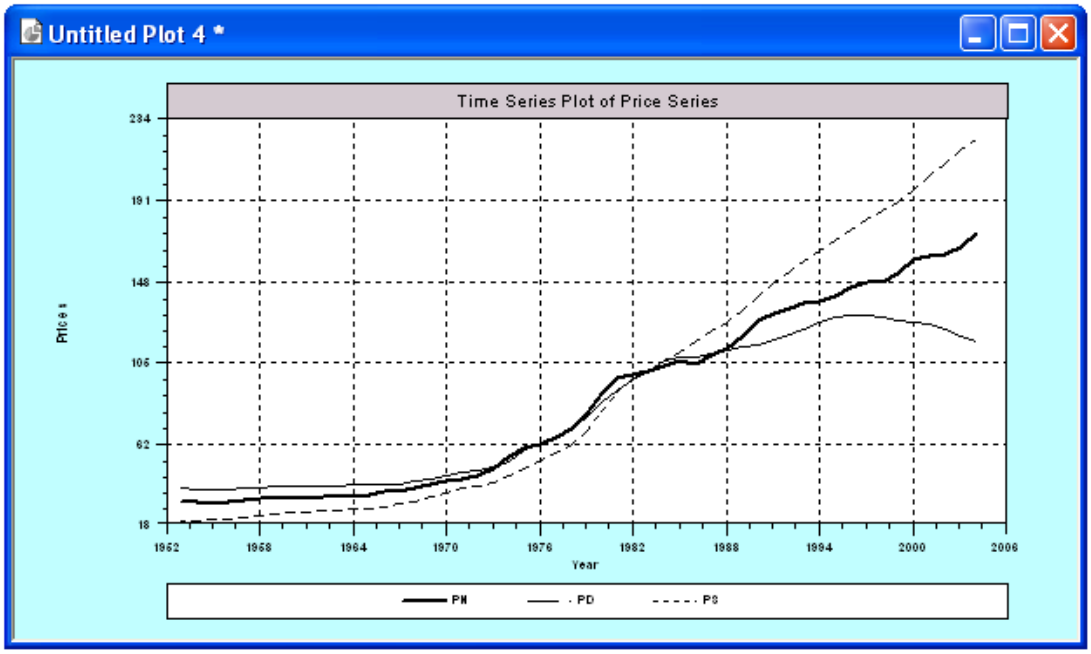


Figure 13.10 Time Series Plot for Several Variables

13.6.7 Plotting Several Variables Against One Variable

To plot several variables against a single one, just include more than one Rhs variable in the command. The command is

```
PLOT      ; Lhs = variable on horizontal axis
          ; Rhs = up to five variables to be plotted
          ; … other options, such as ; Grid and ; Fill $
```

(Note that the command builder dialog box allows you to specify multiple variables.) A different line style is used for each variable if you use **; Fill**. (The time series plot above is an example in which the Lhs variable is the automatically supplied date.) A different type of point is constructed for each if you are using a cross section. Generally, **PLOT** with **; Fill** (see the next section) creates a figure with one or more line plots, joining segments at the points, but suppressing any symbols for the points. The symbols (dots, stars, etc.) may be retained with **; Symbols**. The **; Regression** command is ignored if more than one variable is being plotted. An example is shown in Figure 13.11:

```
PLOT      ; Rhs   = pn, pd, ps
          ; Lhs   = year
          ; Title = Scatter Plot of Price Series
          ; Yaxis = Prices
          ; Grid
          ; Fill
          ; Symbols $
```

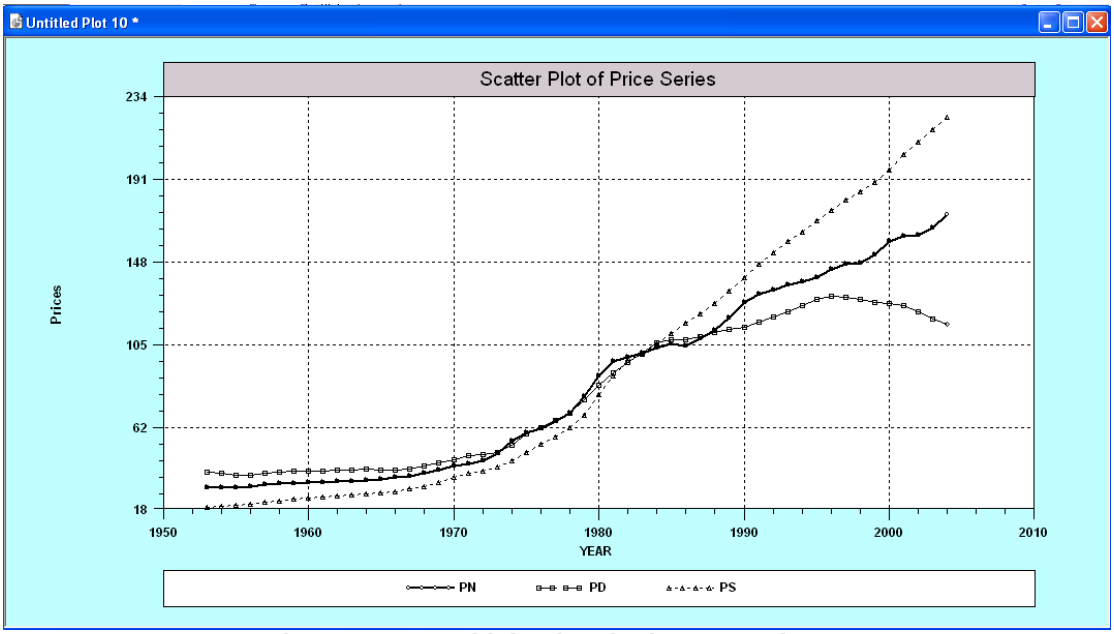


Figure 13.11 Multiple Plots in the Same Figure

NOTE: This is not a time series plot, in spite of the fact that *year* is the variable on the horizontal axis. Although at this point, *LIMDEP* does know that these are time series data, it does not know that '*year*' is a date variable; *year* is just another variable in the data set. If you omit the **; Lhs = variable** specification in the command, *LIMDEP* will label the *x* axis 'YEAR,' but this is not with respect to a variable in your data set; it is the date labeling that you gave in your **DATES** command. Even if you did not have a variable named *year* in your data set, you can obtain a time series style plot with yearly observations, and labeled as such.

13.6.8 Options for Scaling and Labeling the Figure

Scaling

The limits for the vertical and horizontal axes are chosen automatically so that every point appears in the figure. Boundaries are set by the ranges of the variables. You can override these settings, however. The options are as follows:

To set the limits for the horizontal axis use **; Endpoints = lower value, upper value**
To set specific limits for the vertical axis use **; Limits = lower value, upper value**

HINT: If you plot variables of very different magnitudes in the same figure, or if your series has outliers in it, the scaling convention that seeks to include every point in the graph may severely distort your figure. Missing values will also severely distort your scatter plot.

NOTE: If the endpoints or limits that you specify push any points out of the figure – *x* or *y* values are outside the limits – then the specifications are ignored, and the original default values are used.

For example, the top panel in Figure 13.12 is the same as Figure 13.9, produced by the command below without the specification of the endpoints and limits. The lower panel shows the effect of expanding the limits

```
PLOT      ; Rhs    = gasp
          ; Lhs    = g
          ; Title  = Gasoline Consumption vs. Price
          ; Yaxis = Gas_Cons
          ; Grid
          ; Limits = 0,125          ? Set the vertical axis limits
          ; Endpoints = 0,1.2 $     Set the horizontal axis limits
```

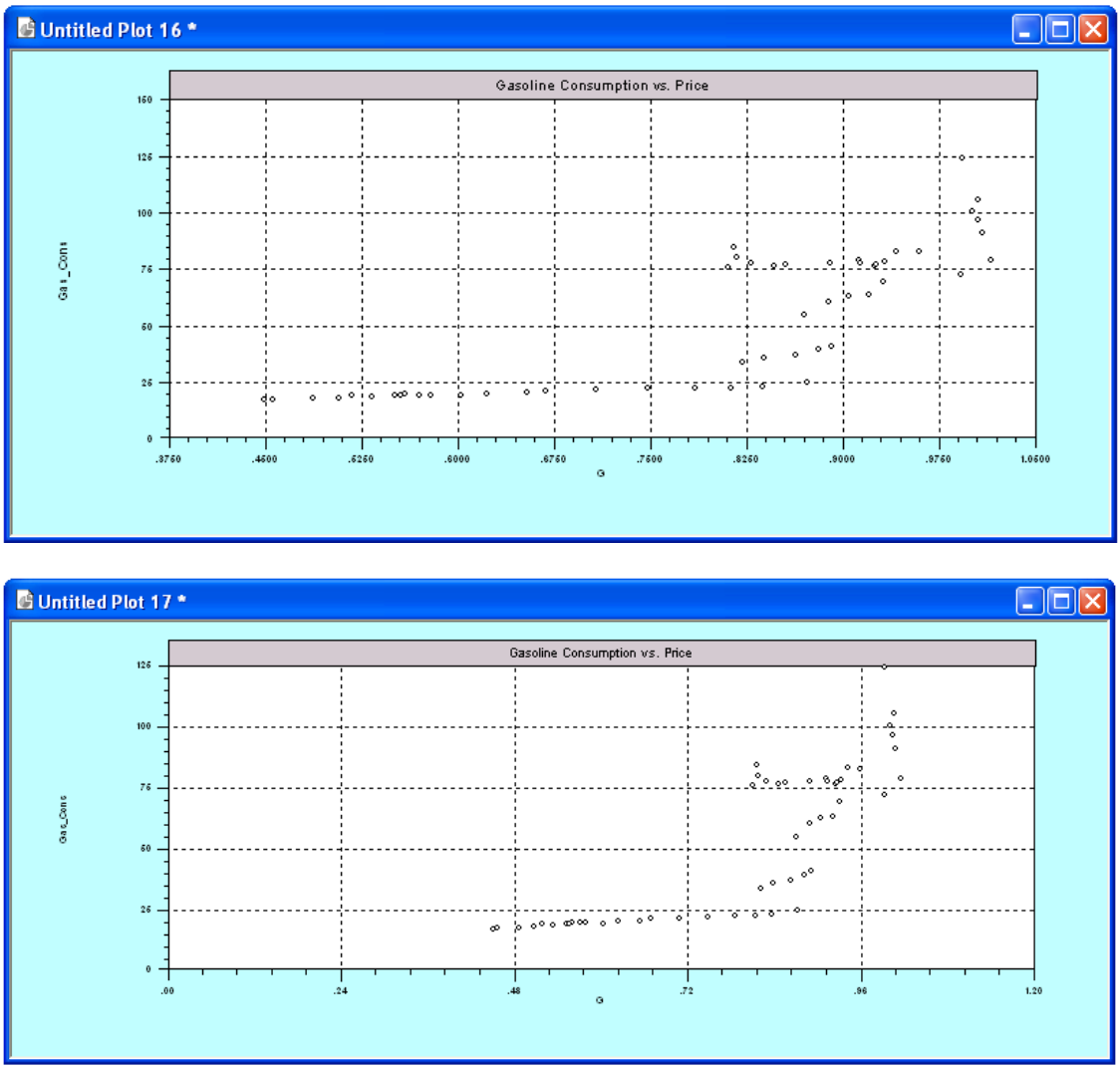


Figure 13.12 Scatter Plots with Rescaled Axes

The following describe some devices for changing the appearance of the figure, and creating particular types of graphs. Some of these have been used in the examples above. More extensive applications appear below.

Grids and Lines in the Plotting Field

It is sometimes helpful when plotting to put a grid in the figure. This makes it easier to relate the points in the graph to the distances on the axes. You may request a grid to be placed in the figure with

; Grid

This divides the screen into a grid of rectangles using dotted bars. The option was used in the preceding examples. You may also put horizontal and/or vertical lines in the figure at specific numerical benchmarks. The syntax is

; Spikes = up to five value(s) to put vertical lines at particular values
; Bars = up to five value(s) to put horizontal lines at particular values

The vertical or horizontal line is drawn from axis to axis, the full width or height of the box. The examples below use these devices to create different types of graphs.

Connecting Points in the Plotting Field

If you are plotting a function or a time series, it may also be useful to connect adjacent points. To do so, add

; Fill

to the command. One way you might use this device would be to draw a function by creating a set of equally spaced values, then plotting the function of these values, connecting the points to create the continuous function.

Chapter 14: The Linear Regression Model

14.1 Introduction

This chapter will detail estimation of the single equation, linear regression model

$$\begin{aligned} y_i &= x_{i1}\beta_1 + x_{i2}\beta_2 + \ldots + x_{iK}\beta_K + \varepsilon_i \\ &= \mathbf{x}_i'\boldsymbol{\beta} + \varepsilon_i,\ i = 1,\ldots,n. \end{aligned}$$

The full set of observations is denoted for present purposes as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

The initial stochastic assumptions are the most restrictive for the linear model:

$$\begin{aligned} E[\varepsilon_i|\,\mathbf{X}\,] &= 0 = E[\varepsilon_i]\ \forall\, i && \text{(zero mean)} \\ \mathrm{Var}[\varepsilon_i\,|\,\mathbf{X}] &= \mathrm{Var}[\varepsilon_i] = \sigma^{2,}\ \forall\, i && \text{(homoscedastic)} \\ \mathrm{Cov}[\varepsilon_i\,,\,\varepsilon_j\,|\,\mathbf{X}] &= \mathrm{Cov}[\varepsilon_i\,,\,\varepsilon_j\,] = 0\ \ \forall\, i\,,j && \text{(nonautocorrelation).} \end{aligned}$$

14.2 Least Squares Regression

The basic command for estimating a linear regression model with least squares is

REGRESS ; Lhs = dependent variable
; Rhs = regressors $

The Rhs list may also include lagged variables and logs of variables. This requests a linear ordinary least squares regression of the Lhs variable on the set of Rhs variables. The standard output from the procedure is listed in the next section.

NOTE: Remember that *LIMDEP* does not automatically include a constant term in the equation. If you want one, be sure to include *one* among the Rhs variables.

An example of the standard output for a linear regression appears below.

```
+----------------------------------------------------+
| Ordinary    least squares regression               |
| Model was estimated Dec 13, 2007 at 08:34:51AM     |
| LHS=LOGG     Mean                 =  -.2571288     |
|              Standard deviation   =   .2384918     |
| WTS=none     Number of observs.   =         52     |
| Model size   Parameters           =          6     |
|              Degrees of freedom   =         46     |
| Residuals    Sum of squares       =   .1070036     |
|              Standard error of e  =   .4823033E-01 |
| Fit          R-squared            =   .9631123     |
|              Adjusted R-squared   =   .9591028     |
| Model test   F[  5,    46] (prob) = 240.21 (.0000) |
| Diagnostic   Log likelihood       =   87.05475     |
|              Restricted(b=0)      =   1.257919     |
|              Chi-sq [  5]  (prob) = 171.59 (.0000) |
| Info criter. LogAmemiya Prd. Crt. =  -5.954335     |
|              Akaike Info. Criter. =  -5.955367     |
| Autocorrel   Durbin-Watson Stat.  =   .2512498     |
|              Rho = cor[e,e(-1)]   =   .8743751     |
+----------------------------------------------------+
+---------+-------------+----------------+--------+---------+----------+
|Variable | Coefficient  | Standard Error |t-ratio |P[|T|>t] | Mean of X|
+---------+-------------+----------------+--------+---------+----------+
 Constant     -11.5997167     1.48817387    -7.795   .0000
 LOGPG         -.03438256      .04201503     -.818   .4174    3.72930296
 LOGINC        1.31596815      .14198287     9.268   .0000    9.67487347
 LOGPNC        -.11963708      .20384305     -.587   .5601    4.38036655
 LOGPUC         .03754405      .09814077      .383   .7038    4.10544880
 LOGPPT        -.21513953      .11655849    -1.846   .0714    4.14194132
```

The statistics reported are as follows:

- The model framework – linear least squares regression
- The date and time when the estimates were computed
- Name of the dependent variable
- Mean of Lhs variable $\bar{y}$ = $(1/n)\Sigma_i y_i$,
- Standard deviation of Lhs variable = $\{[1/(n\text{-}1)]\ \{\Sigma_{i=1}^{N}(y_i - \bar{y})^2\}^{1/2}$
- Name of the weighting variable if one was specified
- Number of observations = n,
- Number of parameters in regression = K,
- Degrees of freedom = n-K,
- Sum of squared residuals $\mathbf{e'e}$ = $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \mathbf{x}_i'\mathbf{b})^2$
- Standard error of e s = $\sqrt{\mathbf{e}'\mathbf{e}/(n-K)}$

- R^2 $R^2 = 1 - \mathbf{e}'\mathbf{e}/\sum_{i=1}^{N}(y_i - \mathbf{x}_i'\mathbf{b})^2$
- Adjusted R^2 $\bar{R}^2 = 1 - (n\text{-}1)/(n\text{-}K)[1 - R^2]$
- F statistic $F[K\text{-}1, n\text{-}K] = [R^2/(K\text{-}1)] / [(1\text{-}R^2)/(n\text{-}K)]$
- Prob value for F $Prob_F = \text{Prob}[F(K\text{-}1, n\text{-}k)] > \text{observed } F$
- Log likelihood $\log L = -n/2[1 + \log 2\pi + \log(\mathbf{e}'\mathbf{e}/n)]$
- Restricted log likelihood $\log L_0 = -n/2[1 + \log 2\pi + \log((1/n)\sum_{i=1}^{n}(y_i - \bar{y})^2)]$
- Chi squared[K-1] $\chi^2 = 2(\log L - \log L_0)$
- Prob value for chi squared $Prob_{\chi 2} = \text{Prob}[\chi^2(K\text{-}1)] > \text{observed chi squared}$
- Log Amemiya Prediction Criterion $= \log[s^2 (1 + K/n)]$
- Akaike Information Criterion $AIC = (\log L - K)/(n/2) - (1 + \log 2\pi)$
- Durbin-Watson $dw = \sum_{t=2}^{T}(e_t - e_{t-1})^2 / \sum_{t=1}^{T} e_t^2$
- Autocorrelation $r = 1 - dw/2$.

The R^2 and related statistics are problematic if your regression does not contain a constant term. For the linear model, *LIMDEP* will check your specification, and issue a warning in the output. Finally, the main table of regression results contains, for each Rhs variable in the regression:

- Name of the variable,
- Coefficient b_k,
- Standard error of coefficient estimate = $se_k = \{[s^2(\mathbf{X}'\mathbf{X})^{-1}]_{kk}\}^{1/2}$.
- t ratio for the coefficient estimate $t_k = b_k / se_k$
- Significance level of each t ratio based on the t distribution with [n-K] degrees of freedom = p value = Prob[$t(n\text{-}K)$] > observed t_k
- Sample mean of the variable.

14.2.1 Retrievable Results

The retrievable results which are saved automatically by the **REGRESS** command are

Matrices:	*b*	= slope vector = $(\mathbf{X'X})^{-1}\ \mathbf{X'y}$
	varb	= estimated covariance matrix = $[\mathbf{e'e}/(n\text{-}K)](\mathbf{X'X})^{-1}$
Scalars:	*ssqrd*	= $\mathbf{e'e}/(n\text{-}K)$
	rsqrd	= R^2
	s	= *s*
	sumsqdev	= sum of squared deviations, $\mathbf{e'e}$
	rho	= autocorrelation coefficient, *r*
	degfrdm	= *n-K*
	sy	= sample standard deviation of Lhs variable
	ybar	= sample mean of Lhs variable
	kreg	= number of independent variables, *K*
	nreg	= number of observations used to compute the regression, *n* Note, this may differ from the sample size if you have skipped observations containing missing values.
	logl	= log likelihood
	exitcode	= 0.0 unless the data were collinear or OLS gives a perfect fit
Last Model:	*b_name*	where the names are the Rhs variables. (See **WALD** in Section R11.5.2)

14.2.2 Predictions and Residuals

To obtain a list of the residuals and fitted values from a linear regression model, add the specification

; List

to the command. The residuals and predicted values may be kept in your data area by using the specifications

; Res = name for residuals

and **; Keep = name for predicted values**

If you are not using the full sample or all of the rows of your data matrix, some of the cells in these columns will be marked as missing. If you have data on the regressors but not the dependent variable, you can use

; Fill

to obtain predictions for the missing data. Remember, though, that the prediction is -999 (missing) for any observation for which any of the *x*s are missing.

The following commands could be used for computing forecast standard errors. This routine uses the matrices *b* (the coefficients) and *varb* (estimated covariance matrix) kept by the regression and scalar *ssqrd* which is s-squared from the regression. The forecast standard errors are the values computed inside the Sqr function in the **CREATE** command.

```
NAMELIST   ; x = the set of regressors  $
REGRESS    ; Lhs = y ; Rhs = x ; Keep = yhat  $
CALC       ; ct = Ttb(.975,degfrdm)  $
CREATE     ; lowerbnd  = yhat -  ct * Sqr(ssqrd + Qfr(x,varb))
           ; upperbnd      = yhat + ct * Sqr(ssqrd + Qfr(x,varb))  $
```

A plot of the residuals from your regression can be requested by adding

```
; Plot
```

to the command. Residuals are plotted against observation number (i.e., simply listed). If you would like to plot them against another variable, change the preceding to

```
; Plot(variable name)
```

These variables can be any existing variables. They need not have been used in the regression. The residuals are sorted according to the variable you name and plotted against it. The plot will show the residuals graphed against either the observation number, the date for time series data, or the variable you specify using the **; Plot(variable)** option described above.

- If there are outliers in the data, this may severely cramp the figure, since the vertical axis is scaled so that every observation will appear.

- The mean residual bar may not appear at zero because the residuals may not have zero mean. They will not if you do not have a constant term in your regression or if you are plotting two stage least squares residuals. Since 0.0 will generally not be the midpoint between the high and low residual, the zero bar will not be in the center of your screen even when you do have a constant term in the model.

14.2.3 Robust Covariance Matrix Estimation

REGRESS will compute robust estimators for the covariance matrix of the least squares estimator for both heteroscedastic and autocorrelated disturbances. Although OLS is generally quite robust, some researchers have advocated other estimators for finite sample purposes. **REGRESS** can also be used to compute the least absolute deviations estimator.

Heteroscedasticity – The White Estimator

For the heteroscedasticity corrected (White) estimator, use

; Heteroscedasticity

in the **REGRESS** command. The White estimator is

$$\text{Est.Var}[\mathbf{b}] = (\mathbf{X}'\mathbf{X})^{-1} \times \sum_{i=1}^{n} e_i^2 \mathbf{x}_i \mathbf{x}_i' \times (\mathbf{X}'\mathbf{X})^{-1}$$

Autocorrelation – The Newey-West Estimator

The Newey-West robust estimator for the covariance matrix of the least squares estimator in the presence of autocorrelation is

$$\text{Est.Var}[\mathbf{b}] = (\mathbf{X}'\mathbf{X})^{-1} \times \sum_{t=1}^{T} e_t^2 \mathbf{x}_t \mathbf{x}_t' \times (\mathbf{X}'\mathbf{X})^{-1}$$
$$+ (\mathbf{X}'\mathbf{X})^{-1} \times \left\{ \frac{1}{T} \sum_{j=1}^{L} \sum_{t=j+1}^{T} \left(1 - \frac{j}{L+1}\right) e_t e_{t-j} \left[\mathbf{x}_t \mathbf{x}_{t-j}' + \mathbf{x}_{t-j} \mathbf{x}_t'\right] \right\} \times (\mathbf{X}'\mathbf{X})^{-1}$$

You (the analyst) must provide the value of L, the number of lags for which the estimator is computed. Then, request this estimator by adding

; Pds = ... the value for L

to the **REGRESS** command.

14.2.4 Restricted Least Squares

This section describes procedures for estimating the restricted regression model,

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}$$
subject to $$\mathbf{R}\boldsymbol{\beta} = \mathbf{q}.$$

R is a $J \times K$ matrix assumed to be of full row rank. That is, we impose J linearly independent restrictions. They may be equality restrictions, inequality restrictions, or a mix of the two. Inequality restricted least squares is handled by a quadratic programming method that is documented in the full manual.

The constrained ordinary least squares estimator is

$$\mathbf{b}_c = \mathbf{b} - (\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1} (\mathbf{Rb}-\mathbf{q})$$

where $$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

is the unrestricted least squares estimator. The estimator of the variance of the constrained estimator is

$$\text{Est.Var}[\mathbf{b}_c] = s^2(\mathbf{X}'\mathbf{X})^{-1} - s^2(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}'[\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}.$$

where $s^2 = (\mathbf{y} - \mathbf{X}\,\mathbf{b}_c)'(\mathbf{y} - \mathbf{X}\,\mathbf{b}_c)/(n-K+J)$.

The parameter vector is written $\mathbf{b} = b_1, b_2, ..., b_K$ where the correspondence is to your Rhs list of variables, including the constant, *one* if it is included. The restrictions are then imposed algebraically with

; CLS: linear function = value ,
linear function = value , ...

For example, the following imposes constant returns to scale (capital coefficient + labor coefficient = 1) on a (hypothetical) production function:

```
CALC       ; Ran (123457) $
SAMPLE     ; 1 – 100 $
CREATE     ; l  = Rnu(1,3)
           ; k = Rnu(.5,2)
           ; y = Exp(3 + .6 * Log(l) + .4 * Log(k) + Rnn(0,4)) $
REGRESS    ; Lhs = Log(y) ; Rhs = one,log(l),log(k)
           ; CLS: b(2) + b(3) = 1 $
```

The following results for restricted least squares are produced. (The **CALC** function sets the seed for the random number generator at a specific value, so you can replicate the results.)

```
+----------------------------------------------------+
| Ordinary    least squares regression               |
| LHS=LOGY     Mean                 =   3.618745     |
|              Standard deviation   =   4.069505     |
| WTS=none     Number of observs.   =        100     |
| Model size   Parameters           =          3     |
|              Degrees of freedom   =         97     |
| Residuals    Sum of squares       =   1624.362     |
|              Standard error of e  =   4.092187     |
| Fit          R-squared            =   .9249439E-02 |
|              Adjusted R-squared   =  -.1117841E-01 |
| Model test   F[  2,    97] (prob) =    .45 (.6372) |
| Diagnostic   Log likelihood       =  -281.2789     |
|              Restricted(b=0)      =  -281.7435     |
|              Chi-sq [  2]  (prob) =    .93 (.6284) |
+----------------------------------------------------+
+---------+-------------+----------------+--------+---------+----------+
|Variable | Coefficient  | Standard Error |t-ratio |P[|T|>t] | Mean of X|
+---------+-------------+----------------+--------+---------+----------+
 Constant      3.20014797     1.06375203     3.008   .0033
 LOGL           .80563172     1.40618551      .573   .5680     .68423500
 LOGK          -.85445443     1.14222281     -.748   .4562     .15523890
```

```
+----------------------------------------------------+
| Linearly restricted regression                     |
| Ordinary    least squares regression               |
| LHS=LOGY     Mean                 =   3.618745     |
|              Standard deviation   =   4.069505     |
| WTS=none     Number of observs.   =        100     |
| Model size   Parameters           =          2     |
|              Degrees of freedom   =         98     |
| Residuals    Sum of squares       =   1629.865     |
|              Standard error of e  =   4.078146     |
| Fit          R-squared            =   .5892627E-02 |
|              Adjusted R-squared   =  -.4251326E-02 |
| Model test   F[  1,    98] (prob) =    .58 (.4478) |
| Diagnostic   Log likelihood       =  -281.4480     |
|              Restricted(b=0)      =  -281.7435     |
|              Chi-sq [  1]  (prob) =    .59 (.4420) |
| Info criter. LogAmemiya Prd. Crt. =   2.831088     |
|              Akaike Info. Criter. =   2.831082     |
| Autocorrel   Durbin-Watson Stat.  =  2.0643771     |
|              Rho = cor[e,e(-1)]   =  -.0321885     |
| Restrictns.  F[  1,    97] (prob) =    .33 (.5678) |
| Not using OLS or no constant. Rsqd & F may be < 0. |
| Note, with restrictions imposed,  Rsqd may be < 0. |
+----------------------------------------------------+
+---------+-------------+----------------+--------+---------+----------+
|Variable | Coefficient  | Standard Error |t-ratio |P[|T|>t] | Mean of X|
+---------+-------------+----------------+--------+---------+----------+
 Constant      2.70416555      .61679283     4.384   .0000
 LOGL          1.43543661      .87473554     1.641   .1040     .68423500
 LOGK          -.43543661      .87473554     -.498   .6198     .15523890
```

You may impose as many restrictions as you wish with this estimator; simply separate the restrictions with commas.

14.2.5 Hypothesis Tests in the Linear Model

The **REGRESS** and **MATRIX** commands can be used to test a variety of hypotheses.

The F statistic for testing the set of J restrictions

$$H_0: \mathbf{R}\boldsymbol{\beta} = \mathbf{q}$$

is

$$F[J,n\text{-}K] =)\mathbf{Rb} - \mathbf{q})'[s^2\mathbf{R}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{R}']^{-1}(\mathbf{Rb} - \mathbf{q})/J$$

$$= [(\mathbf{e}_*'\mathbf{e}_* - \mathbf{e}'\mathbf{e})/J] \,/\, [\mathbf{e}'\mathbf{e}/(n\text{-}K)]$$

where the subscript '*' indicates the sum of squares with the restrictions imposed and b is the unrestricted ordinary least squares estimator. This statistic is included in the diagnostic table whenever you use **; CLS**: to impose linear restrictions.

Consider an example, where now we make two restrictions hold as equalities:

```
SAMPLE    ; 1 - 500 $
CALC      ; Ran(123457) $
CREATE    ; l = Rnu(1,3)  ; k = Rnu(.5,2) ; ll = Log(l) ; lk = Log(k)
          ; lk2 = lk*lk ; ll2 = ll*ll ; lkl = ll*lk
          ; ly = 3 + .6*ll + .4*lk - .05*ll2 - .15*lk2 + .2*lkl + Rnn(0,4) $
REGRESS   ; Lhs = ly ; Rhs = one, ll, lk, ll2, lk2, lkl
          ; CLS:  b(2)+b(3) = 1, b(4)+b(5)+b(6) = 0 $
```

```
+----------------------------------------------------+
| Ordinary    least squares regression               |
| Model was estimated Dec 13, 2007 at 09:30:11AM     |
| LHS=LY       Mean                 =   3.494703     |
|              Standard deviation   =   4.036799     |
| WTS=none     Number of observs.   =        500     |
| Model size   Parameters           =          6     |
|              Degrees of freedom   =        494     |
| Residuals    Sum of squares       =   8108.399     |
|              Standard error of e  =   4.051390     |
| Fit          R-squared            =   .2850337E-02 |
|              Adjusted R-squared   =  -.7242271E-02 |
| Model test   F[  5,   494] (prob) =    .28 (.9227) |
| Diagnostic   Log likelihood       =  -1405.981     |
|              Restricted(b=0)      =  -1406.695     |
|              Chi-sq [  5]  (prob) =   1.43 (.9213) |
| Info criter. LogAmemiya Prd. Crt. =   2.810049     |
|              Akaike Info. Criter. =   2.810048     |
+----------------------------------------------------+
+--------+--------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |t-ratio |P[|T|>t]| Mean of X|
+--------+--------------+----------------+--------+--------+----------+
 Constant|    3.40853548       .76285798     4.468   .0000
 LL      |    -.09422513      2.56561134     -.037   .9707    .66349271
 LK      |    -.05073476      1.16722384     -.043   .9653    .16198512
 LL2     |     .54115572      2.06524209      .262   .7934    .53150388
 LK2     |    -.70218471      1.38385662     -.507   .6121    .16242739
 LKL     |    -.15810258      1.55538131     -.102   .9191    .10542795
```

```
+-----------------------------------------------------+
| Linearly restricted regression                      |
| Ordinary    least squares regression                |
| Model was estimated Dec 13, 2007 at 09:30:11AM      |
| LHS=LY       Mean                 =   3.494703      |
|              Standard deviation   =   4.036799      |
| WTS=none     Number of observs.   =        500      |
| Model size   Parameters           =          4      |
|              Degrees of freedom   =        496      |
| Residuals    Sum of squares       =   8123.764      |
|              Standard error of e  =   4.047043      |
| Fit          R-squared            =   .9607608E-03  |
|              Adjusted R-squared   =  -.5081815E-02  |
| Model test   F[  3,   496] (prob) =    .16 (.9239)  |
| Diagnostic   Log likelihood       =  -1406.454      |
|              Restricted(b=0)      =  -1406.695      |
|              Chi-sq [  3]  (prob) =    .48 (.9231)  |
| Info criter. LogAmemiya Prd. Crt. =   2.803941      |
|              Akaike Info. Criter. =   2.803941      |
| Restrictns.  F[  2,   494] (prob) =    .47 (.6265)  |
| Not using OLS or no constant. Rsqd & F may be < 0.  |
| Note, with restrictions imposed,  Rsqd may be < 0.  |
+-----------------------------------------------------+
+--------+-------------+----------------+--------+--------+----------+
|Variable| Coefficient | Standard Error |t-ratio |P[|T|>t]| Mean of X|
+--------+-------------+----------------+--------+--------+----------+
 Constant|    2.87115229       .26637740    10.779   .0000
 LL      |    1.16811718       .94623345     1.234   .2176    .66349271
 LK      |    -.16811718       .94623345     -.178   .8591    .16198512
 LL2     |    -.33357188      1.05355989     -.317   .7517    .53150388
 LK2     |     .31357011       .76300427      .411   .6813    .16242739
 LKL     |     .02000176      1.26678767      .016   .9874    .10542795
```

14.3 Estimating Models with Heteroscedasticity

For the model in which ω_i is either known or has been estimated already, the weighted least squares estimator is requested with

REGRESS ; Lhs = ... ; Rhs = ... ; Wts = weighting variable $

In computing weighted estimators, we use the formulas:

n = the current sample size, after skipping any missing observations,

$w_i = (n/\Sigma_i W_i)W_i = Scale \times W_i$ (note that $\Sigma_i w_i = n$),

$\mathbf{b}_w = [\Sigma_i w_i \mathbf{x}_i \mathbf{x}_i']^{-1}[\Sigma_i w_i \mathbf{x}_i y_i]$,

$s_w^2 = \Sigma_i w_i (y_i - \mathbf{x}_i'\mathbf{b}_w)^2$,

Est.Var.$[\mathbf{b}_w] = [s_w^2/(n\text{-}K)][\Sigma_i w_i \mathbf{x}_i \mathbf{x}_i']^{-1}$,

where W_i is your weighting variable. Your original weighting variable is not modified (scaled) during this computation. The scale factor is computed separately and carried through the computations.

NOTE: Apart from the scaling, your weighting variable is the reciprocal of the *individual specific variance*, not the standard deviation, and not the reciprocal of the standard deviation. This construction is used to maintain consistency with the other models in *LIMDEP*.

For example, consider the common case, $\text{Var}[\varepsilon_i] = \sigma^2 z_i^2$. For this case, you would use

```
CREATE    ; wt = 1 / z ^ 2 $
REGRESS   ; Lhs = ... ; Rhs = ... ; Wts = wt $
```

14.4 Correcting for First Order Autocorrelation

There are numerous procedures for estimating a linear regression with first order autoregressive disturbances,

$$y_t = \beta' \mathbf{x}_t + \varepsilon_t,$$
$$\varepsilon_t = \rho \varepsilon_{t-1} + u_t .$$

The simplest form of the command is

```
REGRESS   ; Lhs = ... ; Rhs = ... ; AR1
```

The default estimator is the iterative Prais-Winsten algorithm. That is, the first observation is *not* discarded; the full GLS transformation is used. This is a repeated two step estimator:

Step 1. OLS regression of **y** on **X**. Then, estimate ρ with $r = 1 - \frac{1}{2} \times$ Durbin-Watson statistic.

Step 2. OLS regression of $y_1^* = (1 - r^2)^{1/2} y_1$ and $y_t^* = y_t - r y_{t-1}$, $t = 2,...,T$ on the same transformation of $\mathbf{x}_t$.

After Step 2, r is recomputed based on the GLS estimator, and the regression is repeated. This iteration continues until the change in r from one iteration to the next is less than 0.0001. The covariance matrix for the slope estimators is the usual OLS estimator, $s^2(\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}$ based on the transformed data. The asymptotic variance for r is estimated by $(1 - r^2)/(T\text{-}1)$. Results and diagnostics are presented for both transformed and untransformed models.

NOTE: If no other specification is given, the estimator is allowed to iterate to convergence, which usually occurs after a small number of iterations. The updated value of r at each iteration is computed from the Durbin-Watson statistic based on the most recent GLS coefficients estimates. Iterating these estimators to convergence does not produce a maximum likelihood estimator.

Other estimation procedures are requested by adding them to the **; AR1** request:

; AR1 ; Alg = Corc

requests the iterative Cochrane-Orcutt estimator. The first observation is skipped, and the pseudo-difference defined above is applied to the remaining observations. (We do not recommend this estimator, as it needlessly discards the information contained in the first observation, with no accompanying gain in speed, efficiency, or any statistical properties.) Alternatively,

; AR1 ; Alg = MLE

requests the maximum likelihood estimator of Beach and MacKinnon (1978). In this model, the MLE is not GLS because in addition to the generalized sum of squares, the log likelihood function contains an extra term, the Jacobian for the first observation, $\frac{1}{2}\log(1 - \rho^2)$. This term becomes deminimus as $T \to \infty$, so in a large sample, the MLE and the other GLS estimators should not differ substantially.

To use a grid search for the autocorrelation coefficient, use

; AR1 ; Alg = grid(lower, upper, step)

This requests a simple grid search over the indicated range with a stepsize as given. The method used for the grid search is the default Prais-Winsten estimator. To request the Cochrane-Orcutt estimator, instead, use

; AR1 ; Alg = grid(lower, upper, step, 1)

(As before, the Cochrane-Orcutt estimator is inferior to the MLE or Prais-Winsten estimator.) You can request a particular value for ρ by a simple request:

; AR1 ; Rho = specific value

When you use this form of the model command, the output will still contain an estimated standard error for the estimate of ρ, as if it had been estimated. The number of iterations allowed for the first three estimators can be controlled with the specification

; Maxit = maximum

14.5 Two Stage Least Squares

The essential command fitting linear models by instrumental variables is

```
2SLS        ; Lhs = dependent variable
            ; Rhs = list of right hand side variables (all)
            ; Inst =  list of all instrumental variables, including one  $
```

The command for computing instrumental variables or two stage least squares estimates differs from that for ordinary least squares (**REGRESS**) only in a list of instrumental variables. All options are the same as for the linear regression model – see Chapter E5 for details. This includes the specifications of **; AR1** disturbances, **; Plot** for residuals, etc. The list of instruments may include any variables existing in the data set.

HINT: If your equation (Rhs) includes a constant term, *one*, then you should also include *one* in the list of instrumental variables. Indeed, it will often be the case that Inst should include *one* even if the Rhs does not.

Computations use the standard results for two stage least squares. (See, e.g., Greene (2008) There are no degrees of freedom corrections for variance estimators when this estimator is used. All results are asymptotic, and degrees of freedom corrections do not produce unbiased estimators in this context. Thus,

$$\hat{\sigma}^2 = (1/n)\Sigma_i(y_i - \hat{\boldsymbol{\beta}}'\mathbf{x}_i)^2 .$$

This is consistent with most published sources, but (curiously enough) inconsistent with most other commercially available computer programs. It will show up as a proportional difference in all estimated standard errors. If you would prefer that the degrees of freedom correction be made, add the specification

```
; Dfc
```

to your **2SLS** command.

14.5.1 Robust Estimation of the 2SLS Covariance Matrix

The White and Newey-West robust estimators of the covariance matrix of the least squares estimator described in Section 14.2.3 can also be obtained for 2SLS by requesting it in the same fashion. All necessary corrections for the use of the instrumental variables are made in the computation.

14.5.2 Model Output for the 2SLS Command

The output for the **2SLS** command is identical to that for **REGRESS**. The only indication that 2SLS, rather than OLS, was used in estimating the model will be a line at the top of the model results indicating that two stage least squares was used in the computations and a listing of the instrumental variables that will appear above the coefficient estimates. All retrievable results and methods for testing hypotheses are likewise identical.

14.6 Panel Data Models

This chapter will detail estimation of linear models for panel data. The essential structure for most of them is an 'effects' model,

$$y_{it} = \alpha_i + \gamma_t + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it}$$

in which variation across groups (individuals) or time is captured in simple shifts of the regression function – i.e., changes in the intercepts. These models are the *fixed effects* (FE) and *random effects* (RE) models.

The commands for estimation of these models are variants of the basic structure

```
REGRESS     ; Lhs = y ; Rhs = x...
            ; Panel
            ; Str = the name of a stratification variable
   or       ; Pds = specification of the number of periods, variable or fixed
            ; ... other options $
```

14.6.1 Data Arrangement and Setup

Your data are assumed to consist of variables:

$$y_{it}, x_{1it}, x_{2it}, \ldots, x_{Kit}, I_{it}, \quad i = 1,\ldots,N, \; t = 1,\ldots,T_i,$$

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 $t = 1,\ldots,T_i$, on each of $i = 1,\ldots,N$ 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

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.

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

$$\mathbf{X} = \begin{bmatrix} T_1 \text{ observations for group } 1 \\ T_2 \text{ observations for group } 2 \\ \vdots \\ T_N \text{ observations for group } N \end{bmatrix}$$

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.

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.

Specifying the Stratification for Balanced Panels

If you have a balanced panel, that is the same number of observations for each group, you can use

; Pds = T (where you give the specific value of *T*)

to define the panel. There is no need for a stratification variable. For example, in one of our applications, we use a balanced panel data set with $N = 10$ firms and $T = 20$ years. The regression command is

```
REGRESS      ; Lhs = i ; Rhs = one,f,c ; Panel ; Pds = 20 $
```

Specifying the Stratification for Unbalanced Panels

If you will be using unbalanced panels, you may use a stratification variable. The variable is given in the

; Str = group

part of the command. The stratification variable, *group*, may be any indicator that distinguishes the groups – that is, a firm number, any kind of identification number, a telephone number, etc.

For example, the following could be used for our 100 observation panel mentioned earlier

```
REGRESS    ; Lhs = i
           ; Rhs = one,f,c
           ; Str = firm
           ; Panel $
```

Specifying the Stratification with a Count Variable

This and all other panel data estimators in *LIMDEP* use a count variable to specify the stratification in a panel. Your panel is specified with

; Pds = a specific number of periods

as discussed above, or with

; Pds = a variable which gives the group counts

For an example of such a variable, suppose a panel consists of three, then four, then two observations. The nine values taken by the variable, say *ni*, would be 3,3,3,4,4,4,4,2,2. The panel specification would be

; Pds = ni

You may also use this kind of stratification indicator with the linear regression models discussed here.

The Stratification Variable

LIMDEP inspects your stratification variable before estimation of these models begins. If you have specified a balanced panel with **; Pds = T** or if your stratification variable is not a consecutive sequence of integers 1,2,...,*N*, then a new stratification variable called *_stratum* is automatically created for you. The new variable will consist of the consecutive integers. Thereafter, for the same panel, you can use this created variable for your indicator. Suppose, for example, you have a panel data set indicated by postal codes, which are not sorted in your data set (though we assume that all observations for a particular postal code appear together).

Then,

REGRESS ; ... ; Str = postcode $

would automatically create the new stratification variable. Obviously, it is redundant for current purposes, but you might have use for it later. Looking ahead, suppose you were going to fit a linear regression model to one variable in your panel, and a probit model to a different one. For the first one, you can use your postal code to specify the panel. But, the probit model needs a count variable. The next section shows how to obtain this variable.

The Count Variable

In addition to the stratification variable, *_stratum*, this estimator also always creates a count variable named *_groupti*. If you have a balanced panel, the count variable will just equal the number of periods. But, if your panel is unbalanced, *_groupti* is exactly what you will need for the other panel data estimators in *LIMDEP*.

Using REGRESS to Create Stratification and Count Variables

The preceding suggests an extremely helpful tool that should be useful elsewhere in *LIMDEP*, so much so that a special form has been provided for you to use. If you have a stratification variable that identifies the group that an individual is in, you will need a count variable constructed from this in order to use the other panel data estimators in *LIMDEP*. The preceding suggests that computing a panel data based linear regression might be a good way to create the variable, so a method is set up whereby you can use **REGRESS** without actually computing the regression (which you might not have any interest in) to compute this variable. The method is to use the following command:

REGRESS ; Lhs = one ; Rhs = one
; Str = the stratification variable ; Panel $

Notice that this attempts to regress *one* on *one*, which is not a regression at all. *LIMDEP* will notice this, and respond to such a command with results such as the following:

```
=====================================================
| No variables specified. Stratification and count  |
| variables were created. Ngroup =        6         |
| Full sample with missing data  =       20         |
| Stratification variable is       _STRATUM         |
| Count variable for N(i) is       _GROUPTI         |
=====================================================
```

A variable *id* with the results of the command are shown in Figure 14.1.

14.6.2 One Way Fixed and Random Effects Models

The next several sections consider formulation and estimation of one way common effects models,

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

The fixed effects model is

$$\begin{aligned} y_{it} &= \alpha_1 d_{1it} + \alpha_2 d_{2it} + \ldots + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it} \\ &= \alpha_i + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it}, \end{aligned}$$

where $E[\varepsilon_{it}|\mathbf{X}_i] = 0$, $\text{Var}[\varepsilon_{it}|\mathbf{X}_i] = \sigma^2$, $\text{Cov}[\varepsilon_{it},\varepsilon_{js}|\mathbf{X}_i,\mathbf{X}_j] = 0$ for all i,j,

$\text{Cov}[\alpha_i,\mathbf{x}_{it}] \neq \mathbf{0}$.

The efficient estimator for this model in the base case is least squares. The random effects model is

$$y_{it} = \alpha + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it} + u_i$$

where $E[u_i] = 0$, $\text{Var}[u_i] = \sigma_u^2$ $\text{Cov}[\varepsilon_{it},u_i] = 0$.

$\text{Var}[\varepsilon_{it} + u_i] = \sigma^2 = \sigma_\varepsilon^2 + \sigma_u^2$.

Data Editor

3/900 Vars; 3333 Rows: 20 Obs Cell:

	ID	_STRATUM	_GROUPTI
1 »	55	1	3
2 »	55	1	3
3 »	55	1	3
4 »	987	2	3
5 »	987	2	3
6 »	987	2	3
7 »	22	3	2
8 »	22	3	2
9 »	1543	4	6
10 »	1543	4	6
11 »	1543	4	6
12 »	1543	4	6
13 »	1543	4	6
14 »	1543	4	6
15 »	-9	5	3
16 »	-9	5	3
17 »	-9	5	3
18 »	987321	6	3
19 »	987321	6	3
20 »	987321	6	3

Figure 14.1 Internally Created Stratification and Count Variables

For a given *i*, the disturbances in different periods are correlated because of their common component, u_i,

$$\text{Corr}[\varepsilon_{it} + u_i, \varepsilon_{is} + u_i] = \rho = \sigma_u^2 / \sigma^2.$$

The efficient estimator is generalized least squares. Before considering the various different specifications of the two models, we describe two standard devices for distinguishing which seems to be the more appropriate model for a given data set.

Specification Tests for the One Factor Models

Breusch and Pagan's Lagrange multiplier statistic,

$$\text{LM} = \frac{NT}{2(T-1)}\left[\frac{\sum_i \left(\sum_t e_{it}\right)^2}{\sum_i \sum_t e_{it}^2} - 1\right]^2$$

is used to test the null hypothesis that there are no group effects in the random effects model. Arguably, a rejection of the null hypothesis is as likely to be due to the presence of fixed effects. The statistic is computed from the ordinary least squares residuals from a pooled regression. Large values of LM favor the effects model over the classical model with no common effects.

A second statistic is Hausman's chi squared statistic for testing whether the GLS estimator is an appropriate alternative to the LSDV estimator. Computation of the Hausman statistic requires estimates of both the random and fixed effects models. The statistic is

$$H = (\hat{\boldsymbol{\beta}}_{fgls} - \mathbf{b}_{lsdv})'\{\text{Est.Var}[\mathbf{b}_{lsdv}] - \text{Est.Var}[\hat{\boldsymbol{\beta}}_{fgls}]\}^{-1}(\hat{\boldsymbol{\beta}}_{fgls} - \mathbf{b}_{lsdv}).$$

Large values of *H* weigh in favor of the fixed effects model. See Greene (2008) for details. Note that in some data sets, *H* cannot be computed because the difference of the two covariance matrices is not positive definite. (The algebra involved does not guarantee this.) Some authors suggest computing a generalized inverse in such a case, which will force the issue. But, it will not produce an appropriate test statistic. The better strategy in such a case is to take the difference between the two estimators to be random variation, which would favor the random effects estimator.

These two statistics are presented as part of the results when you fit both the fixed effects and random effects models. The **REGRESS** command can be stated to request specifically either of the two. But, if you do not specify either in particular, *LIMDEP* fits both, and near the end of the voluminous output, presents a table such as the following, which contains the LM and H statistics.

```
+---------------------------------------------------+
| Random Effects Model: v(i,t) = e(i,t) + u(i)      |
| Estimates:  Var[e]              =   .322976D+04   |
|             Var[u]              =   .296782D+04   |
|             Corr[v(i,t),v(i,s)] =   .478868       |
| Lagrange Multiplier Test vs. Model (3) =  188.08  | ←
| ( 1 df, prob value =  .000000)                    |
| (High values of LM favor FEM/REM over CR model.)  |
| Fixed vs. Random Effects (Hausman)     =    3.17  | ←
| ( 2 df, prob value =  .205357)                    |
| (High (low) values of H favor FEM (REM).)         |
| Reestimated using GLS coefficients:               |
|  Original:  Var[e]              =   .141468D+05   |
|             Var[u]              =   .325318D+05   |
|             Sum of Squares          .572940D+06   |
|             R-squared               .747284D+00   |
+---------------------------------------------------+
```

HINT: Large values of the Hausman statistic argue in favor of the fixed effects model over the random effects model. Large values of the LM statistic argue in favor of one of the one factor models against the classical regression with no group specific effects. A large value of the LM statistic in the presence of a small Hausman statistic (as in our application) argues in favor of the random effects model.

Commands for One Factor Common Effects Models

The basic command for estimation of the fixed effects model is

```
REGRESS     ; Lhs = dependent variable
            ; Rhs = independent variables
            ; Str = stratification or ; Pds = count (variable)
            ; Panel ; Fixed Effects $
```

The command for random effects is

```
REGRESS     ; Lhs = dependent variable
            ; Rhs = independent variables
            ; Str = stratification or ; Pds = count (variable)
            ; Panel ; Random Effects $
```

If the last line specifies only **; Panel**, and neither fixed nor random effects, then both models are fit. (When you fit a random effects model, the fixed effects regression may be computed in the background without reporting the results, in order to use the estimates to compute the variance components.)

14.6.3 One Way Fixed Effects Models

The two models estimated with this program are 'one way' or 'one factor' designs of the form

$$y_{it} = \mu_i + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it}$$

where ε_{it} is a classical disturbance with $E[\varepsilon_{it}] = 0$ and $\text{Var}[\varepsilon_{it}] = \sigma_\varepsilon^2$. In the fixed effects model, μ_i is a separate constant term for each unit. Thus, the model may be written

$$\begin{aligned} y_{it} &= \alpha_1 d_{1it} + \alpha_2 d_{2it} + \ldots + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it} \\ &= \alpha_i + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it}, \end{aligned}$$

where the α_is are individual specific constants, and the d_js are group specific dummy variables which equal one only when $j = i$. The fixed effects model is a classical regression model. The complication for the least squares procedure is that N may be very large so that the usual formulas for computing least squares coefficients are cumbersome (or impossible) to apply. The model may be estimated in a simpler form by exploiting the algebra of least squares. This model adds the covariates $\mathbf{x}_{it}$ to the 'model' of the previous section.

Commands for One Way Fixed Effects Models

To invoke this procedure, use the command

REGRESS ; Lhs = y
; Rhs = list of regressors
; Str = stratification variable or ; Pds = number of periods
; Panel
; Fixed Effects $

You need not include *one* among your regressors. The constant is placed in the regression automatically when it is needed. You may also use:

; Output = 2

to list fixed effects in an output file. This will also produce estimated standard errors for the fixed effects. If the number of groups is large, the amount of output can be very large.

Standard options for residuals and fitted values, include the following. All of

; List to display residuals and fitted values
; Keep = name to retain predictions
; Res = name to retain residuals
; Var = a submatrix of the parameter VC matrix
; Fill (missing observations)
; Wts = weighting variable
; Printvc

are available as usual. If your stratification indicators are set up properly for out of sample observations, **; Fill** will allow you to extrapolate outside the estimation sample.

WARNING: If you do not have a stratification indicator already in use, **; Fill** will not work. The *_stratum* variable is set up only for the estimation sample. Thus, with **; Pds = T**, you cannot extrapolate outside the sample.

Two full sets of estimates are computed by this estimator:

1. **Constrained Least Squares Regression:** The fixed effects model above with all of the individual specific constants assumed equal is $y_{it} = \alpha+\boldsymbol{\beta}'\mathbf{x}_{it}+\varepsilon_{it}$. This model is estimated by simple ordinary least squares. It is reported as 'OLS Without Group Dummy Variables.'

2. **Least Squares Dummy Variable:** The fixed effects model with individual specific constant terms is estimated by partitioned ordinary least squares. For the one factor models, we formulate this model with N group specific constants and no overall constant.

NOTE: *If you have time invariant regressors, such as sex or region, you cannot compute the fixed effects estimator*. (The fixed effects estimator requires that there be within group variation in all variables for at least some groups.) In this case, you should use **; Random** in your command.

Program Output for One Way Fixed Effects Models

For purposes of the discussion, define the four models:

Model 1	$y_{it} = \alpha + \varepsilon_{it}$		(no group effects or xs),
Model 2	$y_{it} = \alpha_i + \varepsilon_{it}$		(group dummies only),
Model 3	$y_{it} = \alpha + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it}$		(regressors only),
Model 4	$y_{it} = \alpha_i + \boldsymbol{\beta}'\mathbf{x}_{it} + \varepsilon_{it}$		(xs and group effects).

Output from this program, in the order in which it will appear, is as follows:

1. Ordinary least squares regression of y on a single constant and the regressors, $x_1,...,x_K$. *These K variables do not include one*. This is Model 3 above. Output consists of the standard results for least squares regression. The diagnostic statistics in this regression output will also include the unconditional analysis of variance for the dependent variable. This is the usual ANOVA for the groups, ignoring the regressors. (See Section E11.5.1 for details.) The output from this procedure could be used to test the hypothesis that the unconditional mean of y is the same in all groups. (This test is done by the program. See part 3 below.)

2. Ordinary least squares estimates of Model 4 above. Output is the same as in part 1, the usual for a least squares regression. The estimates of the dummy variable coefficients and the estimated standard errors are listed in the output file if requested with **; Output = 2**. (There may be hundreds or thousands of them!)

3. Test statistics for the various classical models. The table contains

   a. For Models 1 - 4, the log likelihood function, sum of squared residuals based on the least squares estimates, and R^2.

   b. Chi squared statistics based on the likelihood functions and F statistics based on the sums of squares for testing the restrictions of:

      - Model 1 as a restriction on Model 2 (no group effects on the mean of *y*),
      - Model 1 as a restriction on Model 3 (no fit in the regression of *y* on *x*s),
      - Model 1 as a restriction on Model 4 (no group effects or fit in regression),
      - Model 2 as a restriction on Model 4 (group effects but no fit in regression),
      - Model 3 as a restriction on Model 4 (fit in regression but no group effects).

      The statistic, degrees of freedom, and prob value (probability that the statistic would be equaled or exceeded by the chi squared or F random variable) are given for each hypothesis.

Saved Results

As always, the matrices *b* and *varb* are saved by the procedure. In addition, a matrix *alphafe* will contain the estimates of the fixed effects, α_i. This matrix is limited to 20,000 cells, so if your data have more than 20,000 groups, *alphafe* will contain the first 20,000 fixed effects computed. Estimates of the variances or standard errors of the fixed effects are not kept. But, a simple method of computing them is given below.

Scalars that are kept are:

ssqrd	= s^2 from least squares dummy variable (LSDV)
rsqrd	= R^2 from LSDV
s	= $\sqrt{s^2}$ from LSDV
sumsqdev	= sum of squared residuals from LSDV
rho	= estimated disturbance autocorrelation from whatever model is fit last
degfrdm	= $\Sigma_i T_i$ - K
sy	= standard deviation of Lhs variable
ybar	= mean of Lhs variable
kreg	= K
nreg	= total number observations
logl	= log likelihood from LSDV model
exitcode	= 0.0 if the model was estimable
ngroup	= number of groups
nperiod	= number of periods. This will be 0.0 if you fit a one way model.

The *Last Model* is constructed as usual, *b_variable*. Predicted values are based on the last model estimated, one or two way, fixed or random. Predictions are not listed when you use the group means estimator, but they can be computed with **MATRIX**.

Robust Estimation of the Fixed Effects OLS Covariance Matrix

There is a counterpart to the White estimator for unspecified heteroscedasticity for the one way fixed effects model. The model is

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

Suppose that every ε_{it} has a different variance, σ_{it}^2 In the fashion of White's estimator for the linear model, the natural approach is simply to replace ε_{it}^2 with e_{it}^2 in the preceding, and compute

$$\text{Est.Asy.Var}[\mathbf{b}_{lsdv}] = [\mathbf{X}^{*\prime}\mathbf{X}^*]^{-1}\,\mathbf{X}^{*\prime}\hat{\boldsymbol{\Omega}}\,\mathbf{X}^*\,[\mathbf{X}^{*\prime}\mathbf{X}^*]^{-1}$$

where '*' denotes deviations from group means. This produces the same results as if the White correction were applied to the OLS results in full model including both regressors and group dummy variables.

If the variances can be assumed to be the same for all observations in the *i*th group, then each group specific variance can be estimated by the group mean squared residual, and the result inserted directly into the formulas above. In this case, $\boldsymbol{\Omega}$ becomes a block diagonal matrix, in which the *i*th diagonal block is $\sigma_i^2\mathbf{I}$. (This resembles the time series/cross section model discussed at the end of this chapter.) In practical terms, we simply replace e_{it}^2 with s_i^2 in the estimate of the asymptotic covariance matrix. To request these estimators, add

; Het or **; Het ; Hc1** or **; Het ; Hc2** or **; Het ; Hc3**

and **; Het = group**

respectively to the **REGRESS** command.

The asymptotic covariance matrix for the fixed effects estimator may also be estimated with a Newey-West style correction for autocorrelation. Request this computation with

; Lags = the number of lags, up to 10

14.6.4 One Way Random Effects Model

The command for the random effects model is

```
REGRESS     ; Lhs = dependent variable
            ; Rhs = independent variables
            ; Str = stratification or ; Pds = count (variable)
            ; Panel
            ; Random Effects $
```

This is the base case. Other specifications will be added as we proceed.

NOTE: The default command for panel data regressions is simply

REGRESS ; Lhs = ... ; Rhs = ... ; Panel ; Str =... or ; Pds = ... $

This command requests *both* FE and RE results. The full set of results for both models will be presented. Adding either **; Fixed Effects** or **; Random Effects** will suppress the display of results for the other model.

NOTE: In computing the random effects model, the second step FGLS estimator generally relies on the first step OLS and LSDV (fixed effects) sums of squares. You may be suppressing the FE model, perhaps because of the presence of time invariant variables which preclude the FE model, but not the RE model. In previous versions of *LIMDEP*, and in some other programs, this will force the estimator to rely on another device to estimate the variance components, typically a group means estimator. In the current version of *LIMDEP*, the FE model is computed in the background, whether reported or not. The sums of squares needed are obtainable even in the presence of time invariant variables. Thus, you will get the same results for the RE model whether or not you have allowed *LIMDEP* to report the fixed effects results.

A crucial element of the computation of the random effects model is the estimation of the variance components. You may supply your own values for σ_ε^2 and σ_u^2. The specification is

; Var = s2e,s2u

This overrides all other computations. The values are checked for validity. A nonpositive value forces estimation to halt at that point.

Results Reported by the Random Effects Estimator

After display of any previous results, including ordinary least squares and the fixed effects estimator, a display such as the following:

```
+------------------------------------------------+
| Random Effects Model: v(i,t) = e(i,t) + u(i)   |
| Estimates:  Var[e]              =   .135360D-02  |
|             Var[u]              =   .578177D-02  |
|             Corr[v(i,t),v(i,s)] =   .810298      |
| Lagrange Multiplier Test vs. Model (3) = 3721.10 |
| ( 1 df, prob value =  .000000)                   |
| (High values of LM favor FEM/REM over CR model.) |
| Baltagi-Li form of LM Statistic =        3721.10 |
| Fixed vs. Random Effects (Hausman)     =   37.62 |
| ( 6 df, prob value =  .000001)                   |
| (High (low) values of H favor FEM (REM).)        |
|             Sum of Squares          .843423D+01  |
|             R-squared               .990075D+00  |
+------------------------------------------------+
```

will be presented, followed by the standard form table of coefficient estimates, standard errors, etc.

The results in the table are as follows:

1. Estimates of σ_ε^2 and σ_u^2 based on the least squares dummy variable model residuals. These are used to estimate the variance components. Since there are some potential problems that can arise, the sequence of steps taken in this part is documented in the trace file. The application shows an example. This trace output may be quite lengthy, as several attempts may be made to fit the model with different variance components estimators.

2. The estimate of $\rho = \sigma_u^2 / (\sigma_\varepsilon^2 + \sigma_u^2)$ based on whatever first round estimator has been used.

3. Breusch and Pagan's Lagrange multiplier statistic for testing the REM against the simple linear regression model with no common effects is

$$LM = \frac{(\Sigma_{i=1}^N T_i)^2}{[2\Sigma_{i=1}^N T_i (T_i - 1)]}\left[\frac{\Sigma_{i=1}^N (T_i \bar{e}_i)^2}{\Sigma_{i=1}^N \Sigma_{t=1}^{T_i} e_{it}^2} - 1\right]^2$$
$$= \frac{(\Sigma_{i=1}^N T_i)^2}{[2\Sigma_{i=1}^N T_i (T_i - 1)]}\left[\frac{\Sigma_{i=1}^N [(T_i \bar{e}_i)^2 - \mathbf{e}_i'\mathbf{e}_i]}{\Sigma_{i=1}^N \mathbf{e}_i'\mathbf{e}_i}\right]^2$$

4. Baltagi and Li's modification of the LM statistic for unbalanced panels is also reported with the results. This replaces the term outside the square brackets with

$$B = \frac{\frac{1}{2} N^2 \left(\Sigma_{i=1}^N N / T_i\right)^2}{\Sigma_{i=1}^N \left(1/T_i^2\right) - N\Sigma_{i=1}^N \left(1/T_i\right)}$$

LIMDEP's computation of the LM statistic already accounts for unbalanced panels. The Baltagi and Li modification evidently has some improved finite sample performance. As can be seen in the example above, which uses a balanced panel, the Baltagi and Li correction has no impact when group sizes are all equal. In this case, both these complicated statistics reduce to $NT/[2(T\text{-}1)]$. The prob value is given for the LM statistic. To a degree of approximation, the same result would apply to the Baltagi and Li form.

5. Hausman's chi squared statistic for testing the REM against the FEM is reported next.

$$H = (\hat{\boldsymbol{\beta}}_{FE} - \hat{\boldsymbol{\beta}}_{RE})' \, [Est.Var(\hat{\boldsymbol{\beta}}_{FE}) - Est.Var(\hat{\boldsymbol{\beta}}_{RE})]^{-1} (\hat{\boldsymbol{\beta}}_{FE} - \hat{\boldsymbol{\beta}}_{RE})$$

The prob value and degrees of freedom for the Hausman statistic are reported.

HINT: Large values of the Hausman statistic argue in favor of the fixed effects model over the random effects model. Large values of the LM statistic argue in favor of one of the one factor models against the classical regression with no group specific effects. A large value of the LM statistic in the presence of a large Hausman statistic (as in our application) argues in favor of the fixed effects model.

NOTE: Sometimes it is not possible to compute the Hausman statistic. The difference matrix in the formula above may not be positive definite. The theory does not guarantee this. It is more likely to be so, but still not certain, if the same estimate of σ_ε^2 is used for both cases. As such, *LIMDEP* uses the FGLS estimator of this, however it has been obtained, for the computation. Still, the matrix may fail to be positive definite. In this case, a 0.00 is reported for the statistic and a diagnostic warning appears in the results. Users are warned, some other programs attempt to bypass this issue by using some other matrix or some other device to force a positive statistic. These ad hoc measures do not solve the problem – they merely mask it. At worst, the appropriate zero value can be replaced by a value that appears to be 'significant.'

6. The simple sum of squared residuals based on the random effects coefficients is reported.

7. An R^2 measure is reported (by popular request)

$$R^2 = 1 - \frac{\Sigma_{i=1}^{N}\Sigma_{t=1}^{T_i}(y_{it} - \hat{\boldsymbol{\beta}}'_{RE}\mathbf{x}_{it})^2}{\Sigma_{i=1}^{N}\Sigma_{t=1}^{T_i}(y_{it} - \bar{y})^2}.$$

   Users are warned, this measure can be negative. It is only guaranteed to be positive when OLS has been used to fit a model with a constant term. There are other measures that could be computed, such as the squared correlation between the actual and fitted values, but neither these, nor the one above, are fit measures in the same sense as in the linear model.

8. Estimates of the coefficients, their standard errors, and the ratio of each coefficient to its estimated standard error. (This is asymptotically distributed as standard normal.)

As always, the matrices *b* and *varb* are saved by the procedure. These will be the FGLS estimates of the random effects model

ssqrd	= s^2 from least squares dummy variable (LSDV) estimator or from FGLS
rsqrd	= R^2 from LSDV
s	= $\sqrt{s^2}$ from LSDV
sumsqdev	= sum of squared residuals from LSDV
rho	= estimated disturbance autocorrelation from whatever model is fit last,
degfrdm	= $\Sigma_i T_i$ - K
sy	= standard deviation of Lhs variable
ybar	= mean of Lhs variable
kreg	= K
nreg	= total number observations
logl	= log likelihood from LSDV model
ssqrdu	= estimate of σ_u^2 from FGLS
ssqrde	= estimate of σ_ε^2 from FGLS
ssqrdw	= estimate of σ_w^2 from GLS if two way random effects model is fit
exitcode	= 0.0 if the model was estimable
ngroup	= number of groups

nperiod = number of periods. This will be 0.0 if you fit a one way model.

The *Last Model* is constructed as usual, *b_variable*. Predicted values are based on the last model estimated, one or two way, fixed or random.

Chapter 15: Models for Discrete Choice

15.1 Introduction

We define models in which the response variable being described is inherently discrete as qualitative response (QR) models. This chapter will describe two of *LIMDEP*'s many estimators for qualitative dependent variable model estimators. The simplest of these is the binomial choice models, which are the subject of Section 15.2. The ordered choice model in Section 15.3 is an extension of the binary choice model in which there are more than two ordered, nonquantitative outcomes, such as scores on a preference scale.

15.2 Modeling Binary Choice

A binomial response may be the outcome of a decision or the response to a question in a survey. Consider, for example, survey data which indicate political party choice, mode of transportation, occupation, or choice of location. We model these in terms of probability distributions defined over the set of outcomes. There are a number of interpretations of an underlying data generating process that produce the binary choice models we consider here. All of them are consistent with the models that *LIMDEP* estimates, but the exact interpretation is a function of the modeling framework.

The essential model command for the parametric binary choice models is

PROBIT
or **; Lhs = dependent variable ; Rhs = regressors $**
LOGIT

A latent regression is specified as

$$y^* = \boldsymbol{\beta}'\mathbf{x} + \varepsilon.$$

The observed counterpart to y^* is

$$y = 1 \text{ if and only if } y^* > 0.$$

This is the basis for most of the binary choice models in econometrics, and is described in further detail below. It is the same model as the reduced form in the previous paragraph. Threshold models, such as labor supply and reservation wages lend themselves to this approach.

The probabilities and density functions for the most common binary choice specifications are as follows:

Probit

$$F = \int_{-\infty}^{\beta'\mathbf{x}_i} \frac{\exp(-t^2/2)}{\sqrt{2\pi}} dt = \Phi(\boldsymbol{\beta}'\mathbf{x}_i), \qquad f = \phi(\boldsymbol{\beta}'\mathbf{x}_i)$$

Logit

$$F = \frac{\exp(\boldsymbol{\beta}'\mathbf{x}_i)}{1+\exp(\boldsymbol{\beta}'\mathbf{x}_i)} = \Lambda(\boldsymbol{\beta}'\mathbf{x}_i), \qquad f = \Lambda(\boldsymbol{\beta}'\mathbf{x}_i)[1 - \Lambda(\boldsymbol{\beta}'\mathbf{x}_i)]$$

15.2.1 Model Commands

The model commands for the five binary choice models listed above are largely the same:

PROBIT
or
LOGIT
; Lhs = dependent variable ; Rhs = regressors $

Data on the dependent variable may be either individual or proportions. You need not make any special note of which. *LIMDEP* will inspect the data to determine which type of data you are using. In either case, you provide only a single dependent variable. As usual, you should include a constant term in the model unless your application specifically dictates otherwise.

15.2.2 Output

The binary choice models generate a very large amount of output. Computation begins with least squares estimation in order to obtain starting values.

NOTE: The OLS results will not normally be displayed in the output. To request the display, use **; OLS** in any of the model commands.

Reported Estimates

Final estimates include:

- log$L$ = the log likelihood function at the maximum,
- logL_0 = the log likelihood function assuming all slopes are zero. If your Rhs variables do not include *one*, this statistic will be meaningless. It is computed as

$$\log L_0 = n[P\log P + (1-P)\log(1-P)]$$

where P is the sample proportion of ones.

- The chi squared statistic for testing H_0: $\boldsymbol{\beta} = \mathbf{0}$ (not including the constant) and the significance level = probability that χ^2 exceeds test value. The statistic is

$$\chi^2 = 2(\log L - \log L_0).$$

Numerous other results, listed in detail, will appear with these in the output. The standard statistical results, including coefficient estimates, standard errors, t ratios, and descriptive statistics for the Rhs variables appear next. A complete listing is given below with an example. After the coefficient estimates are given, two additional sets of results appear, an analysis of the model fit and an analysis of the model predictions.

We will illustrate with binary logit and probit estimates of a model for visits to the doctor using the German health care data described in Chapter E2. The first model command is

```
LOGIT     ; Lhs = doctor
          ; Rhs = one,age,hhninc,hhkids,educ,married
          ; OLS $
```

Note that the command requests the optional listing of the OLS starting values. The results for this command are as follows. With the exception of the table noted below, the same results (with different values, of course) will appear for all five parametric models. Some additional optional computations and results will be discussed later.

The initial OLS estimates are generally not reported unless requested with **; OLS**.

```
+-----------------------------------------------+
| Binomial logit model for binary choice        |
| These are the OLS values based on the         |
| binary variables for each outcome Y(i) = j.   |
+-----------------------------------------------+
+--------+-------------+----------------+--------+--------+----------+
|Variable| Coefficient | Standard Error |b/St.Er.|P[|Z|>z]| Mean of X|
+--------+-------------+----------------+--------+--------+----------+
---------+Characteristics in numerator of Prob[Y = 1]
 Constant|     .56661068       .02118790    26.742   .0000
 AGE     |     .00468710       .00029114    16.099   .0000   43.5256898
 HHNINC  |    -.03976003       .01726656    -2.303   .0213    .35208362
 HHKIDS  |    -.05217181       .00680260    -7.669   .0000    .40273000
 EDUC    |    -.01071245       .00131378    -8.154   .0000   11.3206310
 MARRIED |     .01946888       .00757540     2.570   .0102    .75861817
```

Standard results for maximum likelihood estimation appear next (or first if OLS is not presented). These are the results generated for all models fit by maximum likelihood. The Hosmer-Lemeshow chi squared statistic is specific to the binary choice models. It is discussed in Section E18.3.7. The information criteria are computed from the log likelihood, log*L*, and the number of parameters estimated, *K*, as follows:

AIC = Akaike Information Criterion $= -2(\log L - K)/n$

BIC = Bayesian Information Criterion $= -2(\log L - K\log K)/n$

Finite Sample AIC $= -2(\log L - K - K(K+1)/(n-K-1))/n$

HQIC $= -2(\log L - K\log(\log n))/n$

```
Normal exit from iterations. Exit status=0.
+-----------------------------------------+
| Binomial Logit Model for Binary Choice  |
| Maximum Likelihood Estimates            |
| Dependent variable            DOCTOR    |
| Weighting variable              None    |
| Number of observations         27326    |
| Iterations completed               4    |
| Log likelihood function       -17673.10 |
| Number of parameters               6    |
| Info. Criterion: AIC =          1.29394 |
|   Finite Sample: AIC =          1.29394 |
| Info. Criterion: BIC =          1.29574 |
| Info. Criterion:HQIC =          1.29452 |
| Restricted log likelihood     -18019.55 |
| McFadden Pseudo R-squared      .0192266 |
| Chi squared                   692.9077  |
| Degrees of freedom                 5    |
| Prob[ChiSqd > value] =         .0000000 |
| Hosmer-Lemeshow chi-squared = 110.37153 |
| P-value=  .00000 with deg.fr. =      8  |
+-----------------------------------------+
+--------+-------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |b/St.Er.|P[|Z|>z]| Mean of X|
+--------+-------------+----------------+--------+--------+----------+
---------+Characteristics in numerator of Prob[Y = 1]
 Constant|     .25111543       .09113537     2.755   .0059
 AGE     |     .02070863       .00128517    16.114   .0000   43.5256898
 HHNINC  |    -.18592232       .07506403    -2.477   .0133    .35208362
 HHKIDS  |    -.22947000       .02953694    -7.769   .0000    .40273000
 EDUC    |    -.04558783       .00564646    -8.074   .0000   11.3206310
 MARRIED |     .08529305       .03328573     2.562   .0104    .75861817
```

The next set of results computes various fit measures for the model. This table of information statistics is produced only for the logit model. It is generally used for analysis of the generalized maximum entropy (GME) estimator of the multinomial logit model, but it also provides some useful information for the binomial model even when fit by ML instead of GME. The entropy statistics are computed as follows:

$$Entropy \;=\; -\,\Sigma_i\, P_i \log P_i$$

where P_i is the probability predicted by the model. The three 'models' are '*M*,' the model fit by maximum likelihood, '*MC*,' the model in which all predicted probabilities are the sample proportion of ones (here 0.6291), and '*M*0,' (no model) in which all predicted probabilities are 0.5. The normalized entropy is the entropy divided by *n*log2. Finally, the entropy ratio statistic equals 2(*n*log2)(1 - normalized entropy). The percent correct predicted values are discussed below.

The next set of results examines the success of the prediction rule

$$\text{Predict } y_i = 1 \text{ if } P_i \;>\; P^* \text{ and 0 otherwise}$$

where P^* is a defined threshold probability. The default value of P^* is 0.5, which makes the prediction rule equivalent to 'Predict $y_i = 1$ if the model says the predicted event $y_i = 1 \mid \mathbf{x}_i$ is more likely than the complement, $y_i = 0 \mid \mathbf{x}_i$.' You can change the threshold from 0.5 to some other value with

; Limit = your *P**

```
+--------------------------------------------------------------------+
| Information Statistics for Discrete Choice Model.                  |
|                            M=Model MC=Constants Only   M0=No Model |
| Criterion F (log L)   -17673.09788      -18019.55173  -18940.93986 |
| LR Statistic vs. MC      692.90772            .00000        .00000 |
| Degrees of Freedom         5.00000            .00000        .00000 |
| Prob. Value for LR          .00000            .00000        .00000 |
| Entropy for probs.     17673.09788       18019.55173   18940.93986 |
| Normalized Entropy          .93306            .95135       1.00000 |
| Entropy Ratio Stat.     2535.68395        1842.77624        .00000 |
| Bayes Info Criterion       1.29537           1.32072       1.38816 |
| BIC(no model) - BIC         .09270            .06744        .00000 |
| Pseudo R-squared            .01923            .00000        .00000 |
| Pct. Correct Pred.        62.85223            .00000      50.00000 |
| Means:       y=0    y=1    y=2    y=3    y=4    y=5     y=6   y>=7 |
| Outcome     .3709  .6291  .0000  .0000  .0000  .0000  .0000  .0000 |
| Pred.Pr     .3709  .6291  .0000  .0000  .0000  .0000  .0000  .0000 |
| Notes: Entropy computed as Sum(i)Sum(j)Pfit(i,j)*logPfit(i,j).     |
|        Normalized entropy is computed against M0.                  |
|        Entropy ratio statistic is computed against M0.             |
|        BIC = 2*criterion - log(N)*degrees of freedom.              |
|        If the model has only constants or if it has no constants,  |
|        the statistics reported here are not useable.               |
+--------------------------------------------------------------------+
```

A variety of fit measures for the model are listed.

```
+----------------------------------------+
| Fit Measures for Binomial Choice Model |
| Logit    model for variable DOCTOR     |
+----------------------------------------+
| Proportions P0= .370892   P1= .629108  |
| N =   27326 N0=   10135   N1=   17191  |
| LogL=   -17673.098 LogL0=  -18019.552  |
| Estrella = 1-(L/L0)^(-2L0/n) = .02528  |
+----------------------------------------+
|     Efron |  McFadden  |  Ben./Lerman  |
|    .02435 |    .01923  |       .54487  |
|    Cramer | Veall/Zim. |     Rsqrd_ML  |
|    .02470 |    .04348  |       .02504  |
+----------------------------------------+
| Information  Akaike I.C. Schwarz I.C.  |
| Criteria        1.29394       1.29574  |
+----------------------------------------+
+---------------------------------------------------------+
|Predictions for Binary Choice Model.  Predicted value is |
|1 when probability is greater than  .500000, 0 otherwise.|
|Note, column or row total percentages may not sum to     |
|100% because of rounding. Percentages are of full sample.|
+------+----------------------------------+---------------+
|Actual|         Predicted Value          |               |
|Value |       0               1        | Total Actual  |
+------+---------------+----------------+---------------+
|  0   |    378 (  1.4%)|   9757 ( 35.7%)|  10135 ( 37.1%)|
|  1   |    394 (  1.4%)|  16797 ( 61.5%)|  17191 ( 62.9%)|
+------+---------------+----------------+---------------+
|Total |    772 (  2.8%)|  26554 ( 97.2%)|  27326 (100.0%)|
+------+---------------+----------------+---------------+
```

This table computes a variety of conditional and marginal proportions based on the results using the defined prediction rule. For examples, the 97.708% equals (16797/17191)100% while the 63.256% is (16797/26554)100%.

```
========================================================================
Analysis of Binary Choice Model Predictions Based on Threshold =  .5000
------------------------------------------------------------------------
Prediction Success
------------------------------------------------------------------------
Sensitivity = actual 1s correctly predicted                     97.708%
Specificity = actual 0s correctly predicted                      3.730%
Positive predictive value = predicted 1s that were actual 1s    63.256%
Negative predictive value = predicted 0s that were actual 0s    48.964%
Correct prediction = actual 1s and 0s correctly predicted       62.852%
------------------------------------------------------------------------
Prediction Failure
------------------------------------------------------------------------
False pos. for true neg. = actual 0s predicted as 1s            96.270%
False neg. for true pos. = actual 1s predicted as 0s             2.292%
False pos. for predicted pos. = predicted 1s actual 0s          36.744%
False neg. for predicted neg. = predicted 0s actual 1s          51.036%
False predictions = actual 1s and 0s incorrectly predicted      37.148%
========================================================================
```

Retained Results

The results saved by the binary choice models are:

Matrices:	*b*	=	estimate of β (also contains γ for the Burr model)
	varb	=	asymptotic covariance matrix
Scalars:	*kreg*	=	number of variables in Rhs
	nreg	=	number of observations
	logl	=	log likelihood function

15.2.3 Analysis of Marginal Effects

Marginal effects in a binary choice model may be obtained as

$$\frac{\partial E[y\,|\,\mathbf{x}]}{\partial \mathbf{x}} = \frac{\partial F(\boldsymbol{\beta}'\mathbf{x})}{\partial \mathbf{x}} = \frac{dF(\boldsymbol{\beta}'\mathbf{x})}{d(\boldsymbol{\beta}'\mathbf{x})}\boldsymbol{\beta} = F'(\boldsymbol{\beta}'\mathbf{x})\boldsymbol{\beta} = f(\boldsymbol{\beta}'\mathbf{x})\boldsymbol{\beta}$$

That is, the vector of marginal effects is a scalar multiple of the coefficient vector. The scale factor, $f(\beta'x)$, is the density function, which is a function of **x**. (The densities for the five binary choice models are listed in Section E18.3.1.) This function can be computed at any data vector desired. You can request the computation to be done automatically at the vector of means of the current sample by adding

; Marginal Effects

to your command.

Marginal Effects for Dummy Variables

When one of the variables in **x** is a dummy variable, the derivative approach to estimating the marginal effect is not appropriate. An alternative which is closer to the desired computation for a dummy variable which we denote *z*, is

$$\begin{aligned}\Delta F_z &= \text{Prob}[y = 1 \mid z = 1] - \text{Prob}[y = 1 \mid z = 0] \\ &= F(\boldsymbol{\beta}'\mathbf{x} + \alpha z \mid z = 1) - F(\boldsymbol{\beta}'\mathbf{x} + \alpha z \mid z = 0).\end{aligned}$$

For this type of variable, the asymptotic standard error must be changed as well. This is accomplished simply by changing the appropriate row of **G** to

$$\mathbf{G}_z = [f(\boldsymbol{\beta}'\mathbf{x} + \alpha z)] \begin{pmatrix}\mathbf{x} \\ 1\end{pmatrix}' - [f(\boldsymbol{\beta}'\mathbf{x} + \alpha z)] \begin{pmatrix}\mathbf{x} \\ 0\end{pmatrix}'$$

LIMDEP examines the variables in the model and makes this adjustment automatically.

15.2.4 Robust Covariance Matrix Estimation

The preceding describes a covariance estimator that accounts for a specific, observed aspect of the data. The concept of the 'robust' covariance matrix is that it is meant to account for hypothetical, unobserved failures of the model assumptions. The intent is to produce an asymptotic covariance matrix that is appropriate even if some of the assumptions of the model are not met. (It is an important, but infrequently discussed issue whether the estimator, itself, remains consistent in the presence of these model failures – that is, whether the so called robust covariance matrix estimator is being computed for an inconsistent estimator.) (Section R10.8 in the *Reference Guide* provides general discussion of robust covariance matrix estimation.)

The Sandwich Estimator

It is becoming common in the literature to adjust the estimated asymptotic covariance matrix for possible misspecification in the model which leaves the MLE consistent but the estimated asymptotic covariance matrix incorrectly computed. One example would be a binary choice model with unspecified latent heterogeneity. A frequent adjustment for this case is the 'sandwich estimator,' which is the choice based sampling estimator suggested above with weights equal to one. (This suggests how it could be computed.) The desired matrix is

$$\text{Est.Asy.Var}\left[\hat{\boldsymbol{\beta}}\right] = \left[\sum_{i=1}^{n}\left(\frac{\partial^2 \log F_i}{\partial\hat{\boldsymbol{\beta}}\partial\hat{\boldsymbol{\beta}}'}\right)\right]^{-1}\left[\sum_{i=1}^{n}\left(\frac{\partial \log F_i}{\partial\hat{\boldsymbol{\beta}}}\right)\left(\frac{\partial \log F_i}{\partial\hat{\boldsymbol{\beta}}'}\right)'\right]\left[\sum_{i=1}^{n}\left(\frac{\partial^2 \log F_i}{\partial\hat{\boldsymbol{\beta}}\partial\hat{\boldsymbol{\beta}}'}\right)\right]^{-1}$$

Three ways to obtain this matrix are

```
                ; Wts = one ; Choice Based sampling
or              ; Robust
or              ; Cluster = 1
```

The computation is identical in all cases. (As noted below, the last of them will be slightly larger, as it will be multiplied by $n/(n\text{-}1)$.)

Clustering

A related calculation is used when observations occur in groups which may be correlated. This is rather like a panel; one might use this approach in a random effects kind of setting in which observations have a common latent heterogeneity. The parameter estimator is unchanged in this case, but an adjustment is made to the estimated asymptotic covariance matrix. The calculation is done as follows: Suppose the n observations are assembled in G clusters of observations, in which the number of observations in the ith cluster is n_i. Thus, $\sum_{i=1}^{G} n_i = n$. Let the observation specific gradients and Hessians be

$$\mathbf{g}_{ij} = \frac{\partial \log L_{ij}}{\partial \boldsymbol{\beta}} \quad \mathbf{H}_{ij} = \frac{\partial^2 \log L_{ij}}{\partial \boldsymbol{\beta} \partial \boldsymbol{\beta}'}.$$

The uncorrected estimator of the asymptotic covariance matrix based on the Hessian is

$$\mathbf{V}_H = -\mathbf{H}^{-1} = \left(-\sum_{i=1}^{G} \sum_{j=1}^{n_i} \mathbf{H}_{ij} \right)^{-1}$$

Estimators for some models will use the BHHH estimator, instead;

$$\mathbf{V}_B = \left(\sum_{i=1}^{G} \sum_{j=1}^{n_i} \mathbf{g}_{ij} \mathbf{g}_{ij}' \right)^{-1}$$

Let $\mathbf{V}$ be the estimator chosen. Then, the corrected asymptotic covariance matrix is

$$\text{Est.Asy.Var}\left[\hat{\boldsymbol{\beta}}\right] = \mathbf{V} \frac{G}{G-1} \left[\sum_{i=1}^{G} \left(\sum_{j=1}^{n_i} \mathbf{g}_{ij} \right) \left(\sum_{j=1}^{n_i} \mathbf{g}_{ij} \right)' \right] \mathbf{V}$$

Note that if there is exactly one observation per cluster, then this is $G/(G\text{-}1)$ times the sandwich estimator discussed above. Also, if you have fewer clusters than parameters, then this matrix is singular – it has rank equal to the minimum of G and K, the number of parameters.

To request the estimator, your command must include

; Cluster = specification

where the specification is either the fixed value if all the clusters are the same size, or the name of an identifying variable if the clusters vary in size. Note, this is not the same as the variable in the Pds function that is used to specify a panel. The cluster specification must be an identifying code that is specific to the cluster. For example, our health care data used in our examples is an unbalanced panel. The first variable is a family *id*, which we will use as follows

; Cluster = id

15.3 Ordered Choice Models

The basic ordered choice model is based on the following specification: There is a latent regression,

$$y_i^* = \boldsymbol{\beta}'\mathbf{x}_i + \varepsilon_i,\ \varepsilon_i \sim F(\varepsilon_i|\boldsymbol{\theta}),\ \mathrm{E}[\varepsilon_i|\mathbf{x}_i] = 0,\ \mathrm{Var}[\varepsilon_i|\mathbf{x}_i] = 1,$$

The observation mechanism results from a complete censoring of the latent dependent variable as follows:

$$\begin{aligned} y_i &= 0 \text{ if } y_i \le \mu_0, \\ &= 1 \text{ if } \mu_0 < y_i \le \mu_1, \\ &= 2 \text{ if } \mu_1 < y_i \le \mu_2, \\ &\ldots \\ &= J \text{ if } y_i > \mu_{J-1}. \end{aligned}$$

The latent 'preference' variable, y_i^* is not observed. The observed counterpart to y_i^* is y_i. Four stochastic specifications are provided for the basic model shown above. The *ordered probit* model applies in applications such as surveys, in which the respondent expresses a preference with the above sort of ordinal ranking. The variance of ε_i is assumed to be one, since as long as y_i^*, $\boldsymbol{\beta}$, and ε_i are unobserved, no scaling of the underlying model can be deduced from the observed data. Since the μs are free parameters, there is no significance to the unit distance between the set of observed values of y. They merely provide the coding. Estimates are obtained by maximum likelihood. The probabilities which enter the log likelihood function are

$$\text{Prob}[y_i = j] = \text{Prob}[y_i^* \text{ is in the } j\text{th range}].$$

The model may be estimated either with individual data, with $y_i = 0, 1, 2, \ldots$ or with grouped data, in which case each observation consists of a full set of J+1 proportions, $p_{0i},\ldots,p_{Ji}$.

NOTE: If your data are not coded correctly, this estimator will abort with one of several possible diagnostics – see below for discussion. Your dependent variable must be coded 0,1,...,J. We note that this differs from some other econometric packages which use a different coding convention.

There are numerous variants and extensions of this model which can be estimated: The underlying mathematical forms are shown below, where the CDF is denoted $F(z)$ and the density is $f(z)$. (Familiar synonyms are given as well.)

Probit: $$F(z) = \int_{-\infty}^{z} \frac{\exp(-t^2/2)}{\sqrt{2\pi}} dt = \Phi(z), \quad f(z) = \phi(z)$$

Logit: $$F(z) = \frac{\exp(z)}{1+\exp(z)} = \Lambda(z), \quad f(z) = \Lambda(z)[1 - \Lambda(z)]$$

The *ordered probit* model is an extension of the probit model for a binary outcome with normally distributed disturbances. The *ordered logit model* results from the assumption that ε has a standard logistic distribution instead of a standard normal.

15.3.1 Estimating Ordered Probability Models

The essential command for estimating ordered probability models is

ORDERED ; Lhs = y ; Rhs = regressors $

If you are using individual data, the Lhs variable must be coded 0,1,...,*J*. All the values must be present in the data. *LIMDEP* will look for empty cells. If there are any, estimation is halted. (If value '*j*' is not represented in the data, then the threshold parameter, μ_j is not estimable.) In this circumstance, you will receive a diagnostic such as

```
ORDE,Panel,BIVA PROBIT:A cell has (almost) no observations.
Empty cell: Y        never takes value  2
```

This diagnostic means exactly what it says. The ordered probability model cannot be estimated unless all cells are represented in the data. Users frequently overlook the coding requirement, $y = 0,1,...$ If you have a dependent variable that is coded 1,2,..., you will see the following diagnostic:

```
Models - Insufficient variation in dependent variable.
```

The reason this particular diagnostic shows up is that *LIMDEP* creates a new variable from your dependent variable, say *y*, which equals zero when *y* equals zero, and one when *y* is greater than zero. It then tries to obtain starting values for the model by fitting a regression model to this new variable. If you have miscoded the Lhs variable, the transformed variable always equals one, which explains the diagnostic. In fact, there is no variation in the transformed dependent variable. If this is the case, you can simply use **CREATE** to subtract 1.0 from your dependent variable to use this estimator.

The probit model is the default specification. To estimate an ordered logit, add

; Model = Logit

to the command. The standardized logistic distribution (mean zero, standard deviation approximately 1.81) is used as the basis of the model instead of the standard normal.

15.3.2 Model Structure and Data

This model must include a constant term, *one*, as the first Rhs variable. Since the equation does include a constant term, one of the μs is not identified. We normalize μ_0 to zero. (Consider the special case of the binary probit model with something other than zero as its threshold value. If it contains a constant, this cannot be estimated.) Data may be grouped or individual. (Survey data might logically come in grouped form.) If you provide individual data, the dependent variable is coded 0, 1, 2, ..., *J*. There must be at least three values. Otherwise, the binary probit model applies. If the data are grouped, a full set of proportions, p_0, p_1, ..., p_J, which sum to one at every observation must be provided. In the individual data case, the data are examined to determine the value of *J*, which will be the largest observed value of *y* which appears in the sample. In the grouped data case, *J* is one less than the number of Lhs variables you provide. Once again, we note that other programs sometimes use different normalizations of the model. For example, if the constant term is forced to equal zero, then one will instead, add a nonzero threshold parameter, μ_0, which equals zero in the presence of a nonzero constant term.

15.3.3 Output from the Ordered Probability Estimators

All of the ordered probit/logit models begin with an initial set of least squares results of some sort. These are suppressed unless your command contains **; OLS**. The iterations are then followed by the maximum likelihood estimates in the usual tabular format. The final output includes a listing of the cell frequencies for the outcomes. When the data are stratified, this output will also include a table of the frequencies in the strata. The log likelihood function, and a log likelihood computed assuming all slopes are zero are computed. For the latter, the threshold parameters are still allowed to vary freely, so the model is simply one which assigns each cell a predicted probability equal to the sample proportion. This appropriately measures the contribution of the nonconstant regressors to the log likelihood function. As such, the chi squared statistic given is a valid test statistic for the hypothesis that all slopes on the nonconstant regressors are zero.

The sample below shows the standard output for a model with six outcomes. These are the German health care data described in detail in Chapter E2. The dependent variable is the self reported health satisfaction rating. For the purpose of a convenient sample application, we have truncated the health satisfaction variable at five by discarding observations – in the original data set, it is coded 0,1,...,10.

```
+---------------------------------------------+
| Ordered Probability Model                   |
| Maximum Likelihood Estimates                |
| Dependent variable              NEWHSAT     |
| Weighting variable                 None     |
| Number of observations             8140     |
| Log likelihood function       -11284.69     |
| Number of parameters                  9     |
| Info. Criterion: AIC =          2.77486     |
| Info. Criterion: BIC =          2.78261     |
| Restricted log likelihood     -11308.02     |
| McFadden Pseudo R-squared      .0020635     |
| Chi squared                   46.66728     |
| Degrees of freedom                    4     |
| Prob[ChiSqd > value] =         .0000000     |
| Underlying probabilities based on Normal    |
|    Cell frequencies for outcomes            |
|  Y Count Freq  Y Count Freq  Y Count Freq   |
|  0   447 .054  1   255 .031  2   642 .078   |
|  3  1173 .144  4  1390 .170  5  4233 .520   |
+---------------------------------------------+
+--------+-------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |b/St.Er.|P[|Z|>z]| Mean of X|
+--------+-------------+----------------+--------+--------+----------+
---------+Index function for probability
 Constant|    1.32892012       .07275667    18.265   .0000
 FEMALE  |     .04525825       .02546350     1.777   .0755    .52936118
 HHNINC  |     .35589979       .07831928     4.544   .0000    .32998942
 HHKIDS  |     .10603682       .02664775     3.979   .0001    .33169533
 EDUC    |     .00927669       .00629721     1.473   .1407   10.8759203
---------+Threshold parameters for index
 Mu(1)   |     .23634786       .01236704    19.111   .0000
 Mu(2)   |     .62954428       .01439990    43.719   .0000
 Mu(3)   |    1.10763798       .01405938    78.783   .0000
 Mu(4)   |    1.55676227       .01527126   101.941   .0000
```

The model output is followed by a $(J+1)\times(J+1)$ frequency table of predicted versus actual values. (This table is not given when data are grouped or when there are more than 10 outcomes.) The predicted outcome for this tabulation is the one with the largest predicted probability.

```
+-------------------------------------------------------------------------+
|   Cross tabulation of predictions. Row is actual, column is predicted.  |
|   Model = Probit    .  Prediction is number of the most probable cell.  |
+-------+-------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Actual|Row Sum|  0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  |
+-------+-------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|      0|    447|    0|    0|    0|    0|    0|  447|
|      1|    255|    0|    0|    0|    0|    0|  255|
|      2|    642|    0|    0|    0|    0|    0|  642|
|      3|   1173|    0|    0|    0|    0|    0| 1173|
|      4|   1390|    0|    0|    0|    0|    0| 1390|
|      5|   4233|    0|    0|    0|    0|    0| 4233|
+-------+-------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|Col Sum|   8140|    0|    0|    0|    0|    0| 8140|    0|    0|    0|    0|
+-------+-------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

Even though the model appears to be highly significant, the table of predictions has some large gaps in it. The estimation criterion for the ordered probability model is unrelated to its ability to predict those cells, and you will rarely see a predictions table that closely matches the actual outcomes. It often happens that even in a set of results with highly significant coefficients, only one or a few of the outcomes are predicted by the model.

Computation of predictions and ancillary variables is as follows: For each observation, the predicted probabilities for all $J+1$ outcomes are computed. Then if you request **; List**, the listing will contain

Predicted Y is the *Y* with the largest probability.

Residual is the largest of the $J+1$ probabilities (i.e., Prob[y = fitted *Y*]).

Var1 is the estimate of $E[y_i] = \sum_{i=0}^{J} i \times \text{Prob}[Y_i = i]$.

(Note that since the outcomes are only ordinal, this is not a true expected value.

Var2 is the probability estimated for the observed *Y*.

Estimation results kept by the estimator are as follows:

Matrices: *b* = estimate of β,
varb = estimated asymptotic covariance,
mu = J-1 estimated μs.

Scalars: *kreg*, *nreg*, and *logl*.

Last Model: The labels are *b_variables*, *mu*1, ...

The specification **; Par** adds μ (the set of estimated threshold values) to *b* and *varb*. The additional matrix, *mu* is kept regardless, but the estimated asymptotic covariance matrix is lost unless the command contains **; Par**.

15.3.4 Marginal Effects

Marginal effects in the ordered probability models are quite involved. Since there is no meaningful conditional mean function to manipulate, we consider, instead, the effects of changes in the covariates on the cell probabilities. These are:

$$\partial \text{Prob}[\text{cell } j]/\partial \mathbf{x}_i = [f(\mu_{j-1} - \boldsymbol{\beta}'\mathbf{x}_i) - f(\mu_j - \boldsymbol{\beta}'\mathbf{x}_i)] \times \boldsymbol{\beta},$$

where $f(.)$ is the appropriate density for the standard normal, $\phi(\bullet)$, logistic density, $\Lambda(\bullet)(1-\Lambda(\bullet))$, Weibull or Gompertz. Each vector is a multiple of the coefficient vector. But it is worth noting that the magnitudes are likely to be very different. In at least one case, Prob[cell 0], and probably more if there are more than three outcomes, the partial effects have exactly the opposite signs from the estimated coefficients. Thus, in this model, it is important to consider carefully the interpretation of the coefficient estimates. Marginal effects for all cells can be requested by including **; Marginal Effects** in the command. An example appears below.

NOTE: This estimator segregates dummy variables for separate computation in the marginal effects. The marginal effect for a dummy variable is the simple difference of the two probabilities, with and without the variable. See the application below for an illustration.

```
+-----------------------------------------------------+
| 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 *.          |
+-----------------------------------------------------+
+--------+--------------+----------------+--------+--------+----------+
|Variable| Coefficient  | Standard Error |b/St.Er.|P[|Z|>z]| Mean of X|
+--------+--------------+----------------+--------+--------+----------+
---------+These are the effects on Prob[Y=00] at means.
 Constant|       .000000    ......(Fixed Parameter).......
 *FEMALE |    -.00498024       .00280960    -1.773   .0763    .52936118
 HHNINC  |    -.03907462       .00862973    -4.528   .0000    .32998942
 *HHKIDS |    -.01131976       .00277405    -4.081   .0000    .33169533
 EDUC    |    -.00101850       .00069179    -1.472   .1409   10.8759203
---------+These are the effects on Prob[Y=01] at means.
 Constant|       .000000    ......(Fixed Parameter).......
 *FEMALE |    -.00209668       .00118069    -1.776   .0758    .52936118
 HHNINC  |    -.01647123       .00362630    -4.542   .0000    .32998942
 *HHKIDS |    -.00483428       .00119623    -4.041   .0001    .33169533
 EDUC    |    -.00042933       .00029148    -1.473   .1408   10.8759203
Effects for Y=02, Y=03 and Y=04 are omitted.
---------+These are the effects on Prob[Y=05] at means.
 Constant|       .000000    ......(Fixed Parameter).......
 *FEMALE |     .01803285       .01014562     1.777   .0755    .52936118
 HHNINC  |     .14180876       .00073836   192.060   .0000    .32998942
 *HHKIDS |     .04218672       .00029837   141.390   .0000    .33169533
 EDUC    |     .00369631       .00250467     1.476   .1400   10.8759203
```

```
+---------------------------------------------------------------------------+
| Summary of Marginal Effects for Ordered Probability Model (probit)        |
+---------------------------------------------------------------------------+
Variable|    Y=00    Y=01    Y=02    Y=03    Y=04    Y=05    Y=06    Y=07 |
---------------------------------------------------------------------------+
ONE          .0000   .0000   .0000   .0000   .0000   .0000
*FEMALE     -.0050  -.0021  -.0041  -.0047  -.0021   .0180
HHNINC      -.0391  -.0165  -.0326  -.0373  -.0164   .1418
*HHKIDS     -.0113  -.0048  -.0096  -.0112  -.0052   .0422
EDUC        -.0010  -.0004  -.0008  -.0010  -.0004   .0037
```

Chapter 16: Censoring and Sample Selection

16.1 Introduction

The models described in this chapter are variations on the following general structure:

Latent Underlying Regression: $y_i^* = \boldsymbol{\beta}'\mathbf{x}_i + \varepsilon_i, \varepsilon_i \sim N[0,\sigma^2]$.
Observed Dependent Variable: if $y_i^* \leq L_i$, then $y_i = L_i$ (lower tail censoring)
if $y_i^* \geq U_i$, then $y_i = U_i$ (upper tail censoring)
if $L_i < y_i^* < U_i$, then $y_i = y_i^* = \boldsymbol{\beta}'\mathbf{x}_i + \varepsilon_i$.

The thresholds, L_i and U_i, may be constants or variables. We accommodate censoring in the upper or lower (or both) tails of the distribution. The most familiar case of this model in the literature is the 'tobit' model, in which $U_i = +\infty$ and $L_i = 0$, i.e., the case in which the observed data contain a cluster of zeros. In the standard 'censored regression,' or tobit model, the censored range of y_i^* is the half of the line below zero. (For convenience, we will drop the observation subscript at this point.) If y^* is not positive, a zero is observed for y, otherwise the observation is y^*. Models of expenditure are typical. We also allow censoring of the upper tail ('on the right'). A model of the demand for tickets to sporting events might be an application, since the actual demand is only observed if it is not more than the capacity of the facility (stadium, etc.). A somewhat more elaborate specification is obtained when the range of y^* is censored in both tails. This is the 'two limit probit' model. An application might be a model of weekly hours worked, in which less than half time is reported as 20 and more than 40 is reported as 'full time,' i.e., 40 or more.

NOTE: The mere presence of a clump of zeros in the data set does not, by itself, adequately motivate the tobit model. The specification of the model also implies that the nonlimit observations will have a continuous distribution with observations near the limit points. In general, if you try to fit a tobit model, e.g., to financial data in which there is a clump of zeros, and the nonzero observations are ordinary financial variables far from zero, the model is as likely as not to break down during estimation. In such a case, the model of sample selection is probably a more appropriate specification.

16.2 Single Equation Tobit Regression Model

The base case considered here is the familiar 'tobit' model:

Latent underlying regression: $y_i^* = \boldsymbol{\beta}'\mathbf{x}_i + \varepsilon_i, \varepsilon_i \sim N[0,\sigma^2]$.
Observed dependent variable: if $y_i^* \leq L_i$, then $y_i = L_i$ (lower tail censoring)
if $y_i^* \geq U_i$, then $y_i = U_i$ (upper tail censoring)
if $L_i < y_i^* < U_i$, then $y_i = y_i^* = \boldsymbol{\beta}'\mathbf{x}_i + \varepsilon_i$.

Within this framework, the most familiar form is the lower censoring only, at zero variant.

16.2.1 Commands

The basic command for estimation of the censored regression, or tobit model is

TOBIT ; Lhs = y ; Rhs = ... $

The default value for the censoring limit is zero, at the left (i.e., the familiar case). Censoring limits can be varied in two fashions. To specify upper, rather than lower tail censoring, add

; Upper

to the model. With no other changes, this would specify a model in which the observed values of the dependent variable would be either zero or negative rather than zero or positive. The specific limit point to use can be changed by using

; Limit = limit value

where '**limit value**' is either a fixed value (number or scalar) or the name of a variable. For example, the model of the demand for sporting events at stadiums with fixed capacities which sell out a significant proportion of the time might be

```
TOBIT     ; Lhs    = tickets
          ; Rhs    = one, price, ...
          ; Upper censoring
          ; Limit  = capacity $
```

Models with censoring in both tails of the distribution are requested by changing the **; Limit** specification to

; Limits = lower limit, upper limit

where '**lower limit**' and **'upper limit**' are either numbers, scalars, or the names of variables (or one of each). For example, in a labor supply model, we might have

; Limits = 20,40

Other options for the tobit model are the standard ones for nonlinear models, including

; Printvc	to display the estimated asymptotic covariance matrix
; List	to display predicted values
; Parameters	to include the estimate of σ in the retained parameter vector
; Maxit = n	to set maximum iterations
; Alg = name	to select algorithm
; Tlf, ; Tlb, ; Tlg	to set the convergence criteria (use **; Set** to keep these settings)
; Output = value	to control the technical output during iterations
; Keep = name	to retain fitted values
; Res = name	to retain residuals
; Marginal Effects	

and so on. Sample clustering for the estimated asymptotic covariance matrix may be requested with

; Cluster = specification

as usual.

16.2.2 Results for the Tobit Model

You may request the display of ordinary least squares results by adding

; OLS

to the command. These will be suppressed if you do not include this request. The OLS values will be used as the starting values for the iterations. Maximum likelihood estimates are presented in full. Note that unlike most of the discrete choice models, there is no restricted log likelihood presented. The maximum likelihood estimates for a model that contains only a constant term are no less complicated than one with covariates, and there is no closed form solution for the (β,σ) parameter pair for this model. For a general test of the joint significance of all the variables in the model, we suggest the standard trio of Neyman-Pearson tests, which can be carried out as follows: First set up the Rhs variables in the model.

```
NAMELIST  ; xvars = the x variables in the model, without the constant term $
CALC      ; kx  = Col(xvars) $
TOBIT     ; Lhs = y ; Rhs = one $
CALC      ; l0 = logl $
```

This command will produce the Lagrange multiplier statistic.

```
TOBIT     ; Lhs = y ; Rhs = xvars,one ; Start = kx_0,b,s ; Maxit = 0 $
TOBIT     ; Lhs = y ; Rhs = xvars,one $
```

Compute the likelihood ratio statistic.

```
CALC      ; List ; lr = 2*(logl - l0) ; 1 - Chi(lr,kx) $
```

This computes a Wald statistic.

```
MATRIX    ; beta = b(1:kx) ; vb = varb(1:kx,1:kx)
          ; List ; Wald = beta'<vb>beta $
CALC      ; List ; 1 - Chi(wald,kx) $
```

Retained output from the model includes

Matrices: *b, varb*

Scalars:

	s	= estimated σ
	ybar, sy, kreg	= number of coefficients,
	nreg	= number of observations
	nonlimts	= number of nonlimit observations in estimating sample

The diagnostic information for the model also includes Lin and Schmidt's LM test for the model specification against the alternative suggested by Cragg as well as a test for nonnormality.

16.2.3 Marginal Effects

The marginal effects in the tobit model when censoring is at the left, at zero, are computed using

$$E[y|\mathbf{x}] = \Phi(\boldsymbol{\beta}'\mathbf{x}/\sigma)[\boldsymbol{\beta}'\mathbf{x} + \sigma\phi(\boldsymbol{\beta}'\mathbf{x}/\sigma)/\Phi(\boldsymbol{\beta}'\mathbf{x}/\sigma)].$$

After some algebra, we find

$$\partial E[y|\mathbf{x}]/\partial\mathbf{x} = \Phi(\boldsymbol{\beta}'\mathbf{x}/\sigma)\boldsymbol{\beta}.$$

The preceding is a broad result which carries over to more general models. That is,

$$\partial E[y|\mathbf{x}]/\partial\mathbf{x} = \text{Prob(nonlimit)}\boldsymbol{\beta}$$

for all specifications of the censoring limits, whether in one tail or both. To obtain a display of the marginal effects for the tobit model, add

; Marginal Effects

to the **TOBIT** command. A full listing of the marginal effects computed at the sample means, including standard errors, the estimated conditional mean, and the scale factor, will be included in the model output.

16.3 Sample Selection Model

Many variants of the 'sample selection' model can be estimated with *LIMDEP*. Most of them share the following structure: A specified model, denoted **A**, applies to the underlying data. However, the observed data are not sampled randomly from this population. Rather, a related variable z^* is such that an observation is drawn from **A** only when z^* crosses some threshold. If the observed data are treated as having been randomly sampled from **A** instead of from the subpopulation of **A** associated with the 'selected' values of z^*, potentially serious biases result. The general solution to the selectivity problem relies upon an auxiliary model of the process generating z^*. Information about this process is incorporated in the estimation of **A**.

Several of the forms of this model which can be estimated with *LIMDEP* depart from Heckman's now canonical form, a linear regression with a binary probit selection criterion model:

$$y = \boldsymbol{\beta}'\mathbf{x} + \varepsilon,$$

$$z^* = \boldsymbol{\alpha}'\mathbf{w} + u,$$

$$\varepsilon, u \sim N[0,0,\sigma_\varepsilon^2, \sigma_u^2, \rho].$$

A bivariate classical (seemingly unrelated) regressions model applies to the structural equations. The standard deviations are σ_ε and σ_u, and the covariance is $\rho\sigma_\varepsilon\sigma_u$. If the data were randomly sampled from this bivariate population, the parameters could be estimated by least squares, or GLS combining the two equations. However, z^* is not observed. Its observed counterpart is z, which is determined by

$$z = 1 \text{ if } z^* > 0$$

and

$$z = 0 \text{ if } z^* \leq 0.$$

Values of y and $\mathbf{x}$ are only observed when z equals one. The essential feature of the model is that under the sampling rule, $E[y|\mathbf{x},z=1]$ is not a linear regression in $\mathbf{x}$, or $\mathbf{x}$ and z. The development below presents estimators for the class of essentially nonlinear models that emerge from this specification.

The basic command structure for the models described in this chapter is

```
PROBIT   ; Lhs = variable z ; Rhs = variables in w ; Hold $
SELECT   ; Lhs = variable y ; Rhs = variables in x $
```

Note that two commands are required for estimation of the sample selection model, one for each structural equation.

16.3.1 Regression Models with Sample Selection

The models described in this section are based on a dichotomous selection mechanism. Heckman's approach to estimation is based on the following observations: In the selected sample,

$$\begin{aligned}
E[y_i|\mathbf{x}_i, \text{in sample}] &= E[y_i \mid \mathbf{x}_i, z_i = 1] \\
&= E[y_i \mid \mathbf{x}_i, \boldsymbol{\alpha}'\mathbf{w}_i + u_i > 0] \\
&= \boldsymbol{\beta}'\mathbf{x}_i + E[\varepsilon_i \mid u_i > -\boldsymbol{\alpha}'\mathbf{w}_i] \\
&= \boldsymbol{\beta}'\mathbf{x}_i + (\rho\sigma_\varepsilon\sigma_u)\{\phi(-\boldsymbol{\alpha}'\mathbf{w}_i)/[1 - \Phi(-\boldsymbol{\alpha}'\mathbf{w}_i)]\} \\
&= \boldsymbol{\beta}'\mathbf{x}_i + (\rho\sigma_\varepsilon\sigma_u)[\phi(\boldsymbol{\alpha}'\mathbf{w}_i)/\Phi(\boldsymbol{\alpha}'\mathbf{w}_i)].
\end{aligned}$$

Given the structure of the model and the nature of the observed data, σ_u cannot be estimated, so it is normalized to 1.0. (We observe the same values of z_i regardless of the value of σ_u.) Then,

$$\begin{aligned}
E[y_i \mid \mathbf{x}_i, \text{in sample}] &= \boldsymbol{\beta}'\mathbf{x}_i + (\rho\sigma_\varepsilon)\lambda_i \\
&= \boldsymbol{\beta}'\mathbf{x}_i + \theta\lambda_i .
\end{aligned}$$

There are some subtle ambiguities in the received applications of this model. First, it is unclear whether the index function, $\boldsymbol{\beta}'\mathbf{x}_i$, or the conditional mean is really the function of interest. If the model is to be used to analyze the behavior of the selected group, then it is the latter. If not, it is unclear. The index function would be of interest if attention were to be applied to the entire population, rather than those expected to be selected. This is application specific. Second, the marginal effects in this model are complicated as well. For the moment, assume that $\mathbf{x}_i$ and $\mathbf{w}_i$ are the same variables. Then,

$$\frac{\partial E[y_i \mid \mathbf{x}_i, z_i = 1]}{\partial \mathbf{x}_i} = \boldsymbol{\beta} + \theta(-\lambda_i \boldsymbol{\alpha}'\mathbf{x}_i - \lambda_i^2)\boldsymbol{\alpha}$$

For any variable x_k which appears in both the selection equation (for z_i) and the regression equation, the marginal effect consists of both the direct part (β_k) and the indirect part, which is of opposite sign – the term in parentheses is always negative; $\theta(-\lambda_i\boldsymbol{\alpha}'\mathbf{x}_i - \lambda_i^2)\alpha_k$. It is not obvious which part will dominate. Most applications have at least some variables that appear in both equations, so this is an important consideration. Note also that variables which do not appear in the index function still affect the conditional mean function through their affect on the inverse Mills ratio (the 'selection variable'). (We note the risk of conflict in the notation used here for the selection term, λ_i, and the loglinear term in the conditional mean functions of the generalized linear models in the previous chapter. There is no relationship between the two. The two uses of 'lambda' are so common in the literature as to have become part of the common parlance and as such, the risk of ambiguity is worse if we try to change the notation used here for clarity.)

LIMDEP contains three estimators for this model, Heckman's two step (or 'Heckit') estimator, full information maximum likelihood, and two step maximum likelihood (which is, more or less, a limited information maximum likelihood estimator). The two step estimator is given here. The others are documented in the full manual for the program.

16.3.2 Two Step Estimation of the Standard Model

Heckman's two step, or 'Heckit' estimation method, is based on the method of moments. It is consistent, but not efficient estimator.

Step 1. Use a probit model for z_i to estimate $\boldsymbol{\alpha}$.
For each observation, compute $\lambda_i = \phi(\boldsymbol{\alpha}'\mathbf{w}_i)/\Phi(\boldsymbol{\alpha}'\mathbf{w}_i)$ using the probit coefficients.

Step 2. Linearly regress y_i on $\mathbf{x}_i$ and λ_i to estimate $\boldsymbol{\beta}$ and $\theta = \rho\sigma_\varepsilon$.
Adjust the standard errors and the estimate of σ_ε^2, which is inconsistent.

The corrected asymptotic covariance matrix for the two step estimator, $(\mathbf{b},c)$, is

$$\text{Asy.Var}[\mathbf{b},c] = \sigma_\varepsilon^2(\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}[\mathbf{X}^{*\prime}(\mathbf{I} - \rho^2\boldsymbol{\Delta})\mathbf{X}^* + \rho^2(\mathbf{X}^{*\prime}\boldsymbol{\Delta}\mathbf{W})\boldsymbol{\Sigma}(\mathbf{W}'\boldsymbol{\Delta}\mathbf{X}^*)](\mathbf{X}^{*\prime}\mathbf{X}^*)^{-1}$$

where $\mathbf{X}^* = [\mathbf{X} : \boldsymbol{\lambda}]$,

$\boldsymbol{\Delta} = \text{diag}[\boldsymbol{\delta}]$,

$\delta_i = -\lambda_i(\boldsymbol{\alpha}'\mathbf{w}_i + \lambda_i)$ $(-1 \le \delta_i \le 0)$,

and $\boldsymbol{\Sigma}$ = asymptotic covariance matrix for the estimator of $\boldsymbol{\alpha}$.

A consistent estimator of σ_ε^2 is $\hat{\sigma}_\varepsilon^2 = \mathbf{e'e}/n - \hat{\theta}^2\bar{\hat{\delta}}$. The remaining parameters are estimated using the least squares coefficients. The computations used in the estimation procedure are those discussed in Heckman (1979) and in Greene (1981).

NOTE: (This is one of our frequently asked questions.) *LIMDEP* always computes the corrected asymptotic covariance matrix, for all variants of selection models in all model frameworks.

The estimator of the correlation coefficient, ρ, is $\text{sign}(\hat{\theta})\sqrt{\hat{\theta}^2/\hat{\sigma}_\varepsilon^2}$. This is the ratio of a regression coefficient (the coefficient on λ_i) and the variance estimator above. Note that it is not a sample moment estimator of the correlation of two variables. This ratio is not guaranteed to be between -1 and +1. (See Greene (1981), which is about this result.) But, note also that an estimate of ρ is needed to compute the asymptotic covariance matrix above, so this is a potential complication. When this occurs, *LIMDEP* uses either +1 or -1, and continues. We emphasize, this is not an error, nor is it a program failure. It is a characteristic of the data. (It may signal some problems with the model.) When this condition occurs, the model results will contain the diagnostic

Estimated correlation is outside the range -1 < r < 1. Using 1.0

This condition is specific to the two step regression estimators. The maximum likelihood estimators discussed below force the coefficient to lie in the unit interval – ρ is estimated directly, not by the method of moments.

To estimate this model with *LIMDEP*, it is necessary first to estimate the probit model, then request the selection model. The pair of commands is

```
PROBIT    ; Lhs = name of z ; Rhs = list for w ; Hold results $
SELECT    ; Lhs = name of y ; Rhs = list for x $
```

For this simplest case, **; Hold ...** may be abbreviated to **; Hold**. All of the earlier discussion for the probit model applies. (See Chapter E18.) This application differs only in the fact the **; Hold** requests that the model specification and results be saved to be used later. Otherwise, they disappear with the next model command. The **PROBIT** command is exactly as described in Chapter E18. The selection model is completely self contained. You do not need to compute or save λ_i.