Trees *package*

Empirical Market Microstructure

(2006, Oxford University Press)

Companion *Mathematica* notebook

Joel Hasbrouck

This package contains routines useful in working with decision trees (as in sequential trade models). The notebook should be stored as as (".m") package. See "TreesDemo.nb" demonstration notebook.

```
BeginPackage["Trees`"];
```

```
Tree::usage = "A Tree object is a multilevel list
    where the entries are expressions corresponding to nodes.";
```

```
ShowTree::usage =
  "ShowTree[t] where t is a list structure representing a decision
    tree. ShowTree displays it as a table that more clearly
    illustrates the branches. ShowTree[{t1,t2,...}] displays an
    array of tree structures. (t1, t2, ... must be conformable).";
```

```
NodePos::usage =
  "NodePos[t,n] where t is a tree (in list form) and n is a list of
    nodes {n1,n2,...,nK}. Returns the positions of nodes
    matching nK that also pass through nodes n1, n2, ...";
```

```
MarkedTree::usage =
  "MarkedTree[t,n] where t is a tree in list form and n is a list
    of nodes {n1, n2, ..., nK}. Displays the tree
    with the selected nodes marked by red stars.";
```

```
CheckConformance::usage =
  "CheckConformance[{t1, t2, ...}] verifies that all of the
    trees have the same list structure.";
```

```
PrTotal::usage =
  "PrTotal[pr] where pr is a tree structure containing the transition
    probabilities. Returns a tree with total probabilities.";
```

```
BuildTree::usage = "BuildTree[TreeLabels,TreePrs] returns
    a list of trees: {labels, transition prs, total prs}.";
```

```
Pr::usage =
  "Pr[tree,n] where tree is a Tree structure and n is a list of nodes,
    {n1, n2, ..., nK} returns the probability of
    terminal nodes nK that pass through n1, n2, ...";
```

```
Begin["`Private`"];
```

```
ShowTree[zz__List] := Module[{z2 = {zz}, t},
   t = GrowTree /@ z2;
   TableForm[Map[
      If[Dimensions[t][[1]] > 1 && ¬ And @@ (" " === # & /@ #),
        FrameBox[TableForm[#]], #] &,
      Transpose[t, {3, 1, 2}], {2}],
     TableAlignments → {Left, Top}, TableSpacing → {.5, .5}] // DisplayForm
  ];
```

```
ShowTree[zz_Tree] := ShowTree @@ List @@ zz;
```

```
NodePos[t_List, n__] := Module[{nn = {n}, k, j},
   k = Position[t, Last[nn]];
   If[Length[nn] === 1, Return[k]];
   nn = Drop[nn, -1];
   Do[
    j = Position[t, {nn[[i]], __}];
    k = Select[k, MemberQ[j, Take[#, Length[First[j]]]] &];,
    {i, Length[nn], 1, -1}
   ];
   k
  ];
```

```
MarkedTree[tree_Tree, n__] := Module[{t},
   t = ReplacePart[BlankTree[tree[[1]]], "★★★", NodePos[tree[[1]], n]];
   Tree[tree[[1]], tree[[2]], tree[[3]], t]];
```

```
MarkedTree[tree_List, n__] :=
  ReplacePart[BlankTree[tree], "★★★", NodePos[tree, n]];
```

```
CheckConformance[t__] := Module[{t2 = {t}, tList},
   If[Length[t2] === 1,
    Print["CheckConformance. Only one tree passed. Nothing to check."];
    Return[True]];
   tList = Table[Position[t2[[i]], {__}], {i, Length[t2]}];
   Do[
    If[
     ¬ SameQ @@ Table[tList[[i, j]], {i, Length[t2]}],
     Print["CheckConformance. First list mismatch at ", j];
     Do[
      Print["Tree ", i, " ", tList[[i, j]]];
      Print[Extract[t2[[i]], Drop[tList[[i, j]], -1]]];,
      {i, Length[t2]}];
     Abort[]
    ],
    {j, Max @@ Length /@ tList}];
   True];
```

```
PrTotal[pr_] := Module[{p = pr},
   Do[p = Replace[p, a_ :↦ Join[{a[[1]]}, a[[1]] * Drop[a, 1]] /; ListQ[a], {i}],
    {i, Depth[pr], 1, -1}];
   p];
```

```
BuildTree[TreeLabels_, TreePrs_, TreeTotalPrs___] := Module[{prt = TreeTotalPrs},
   If[prt == Null, prt = PrTotal[TreePrs]];
   CheckConformance[TreeLabels, TreePrs, prt];
   Tree[TreeLabels, TreePrs, prt]];
```

```
Pr[tree_Tree, n__] := Total @ Extract[tree[[3]], NodePos[tree[[1]], n]];
```

```
AddColumn[m_, zz_: 0] :=
  Transpose[Join[Transpose[m], Table[zz, {1}, {Dimensions[m][[1]]}]]];
AddRow[m_, zz_: 0] := Join[m, Table[zz, {1}, {Dimensions[m][[2]]}]];
AddBranches[m_, branches_List, startRow_, startCol_] :=
  Module[{m2 = m, startRow2 = startRow},
   Do[
    If[Dimensions[m2][[1]] < startRow2, m2 = AddRow[m2, " "]];
    If[Dimensions[m2][[2]] < startCol, m2 = AddColumn[m2, " "]];
    Check[
     If[! ListQ[branches[[i]]],
       m2[[startRow2, startCol]] = branches[[i]];,
       m2[[startRow2, startCol]] = branches[[i, 1]];
       m2 = AddBranches[m2, Drop[branches[[i]], 1], startRow2, startCol + 1];
      ];,
     Print["Error in AddBranches"];
     Print["m2:", TableForm[m2]];
     Print["branches: ", branches];
     Print["Length of branches: ", Length[branches]];
     Print["i: ", i];
     Abort[];
    ];
    startRow2 = Dimensions[m2][[1]] + 1;,
    {i, Length[branches]}
   ];
   m2
  ];
```

```
GrowTree[t_] := AddBranches[{{t[[1]]}}, Drop[t, 1], 1, 2];
```

```
BlankTree[t_] := (t //. a_ :> "" /; FreeQ[a, List]);
```

```
End[];
```

```
EndPackage[];
```