Chapter
3

Classification and Regression Trees

Leland Wilkinson

The TREES module computes classification and regression trees. Classification trees include those models in which the dependent variable (the predicted variable) is categorical. Regression trees include those in which it is continuous. Within these types of trees, the TREES module can use categorical or continuous predictors, depending on whether a CATEGORY statement includes some or all of the predictors.

For any of the models, a variety of loss functions is available. Each loss function is expressed in terms of a goodness-of-fit statistic—the proportion of reduction in error (PRE). For regression trees, this statistic is equivalent to the multiple R^2. Other loss functions include the Gini index, "twoing" (Breiman et al.,1984), and the phi coefficient.

TREES produces graphical trees called **mobiles** (Wilkinson, 1995). At the end of each branch is a density display (box plot, dot plot, histogram, etc.) showing the distribution of observations at that point. The branches balance (like a Calder mobile) at each node so that the branch is level, given the number of observations at each end. The physical analogy is most obvious for dot plots, in which the stacks of dots (one for each observation) balance like marbles in bins.

TREES can also produce a SYSTAT program to code new observations and predict the dependent variable. This program can be saved to a file and run from the command window or submitted as a program file.

Resampling procedures are available in this feature.

Statistical Background

Trees are directed graphs beginning with one node and branching to many. They are fundamental to computer science (data structures), biology (classification), psychology (decision theory), and many other fields. Classification and regression trees are used for prediction. In the last two decades, they have become popular as alternatives to regression, discriminant analysis, and other procedures based on algebraic models. Tree-fitting methods have become so popular that several commercial programs now compete for the attention of market researchers and others looking for software.

Different commercial programs produce different results with the same data, however. Worse, some programs provide no documentation or supporting material to explain their algorithms. The result is a marketplace of competing claims, jargon, and misrepresentation. Reviews of these packages (for example, Levine, 1991; Simon, 1991) use words like "sorcerer," "magic formula," and "wizardry" to describe the algorithms and express frustration at vendors' scant documentation. Some vendors, in turn, have represented tree programs as state-of-the-art "artificial intelligence" procedures capable of discovering hidden relationships and structures in databases.

Despite the marketing hyperbole, most of the now-popular tree-fitting algorithms have been around for decades. The modern commercial packages are mainly microcomputer ports (with attractive interfaces) of the mainframe programs that originally implemented these algorithms. Warnings of abuse of these techniques are not new either (for example, Einhorn, 1972; Bishop et al.,1975). Originally proposed as automatic procedures for detecting interactions among variables, tree-fitting methods are actually closely related to classical cluster analysis (Hartigan, 1975).

This introduction will attempt to sort out some of the differences between algorithms and illustrate their use on real data. In addition, tree analyses will be compared to discriminant analysis and regression.

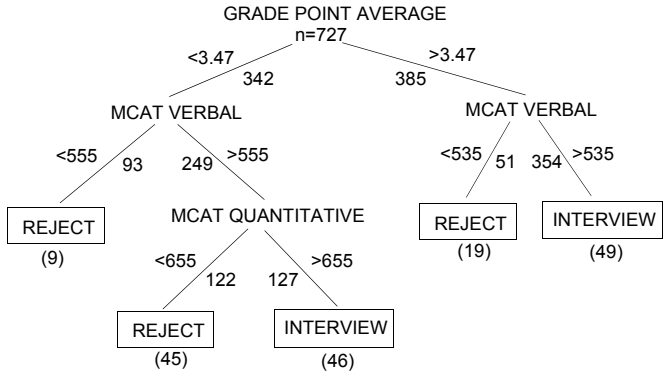
The Basic Tree Model

The figure below shows a tree for predicting decisions by a medical school admissions committee (Milstein et al., 1975). It was based on data for a sample of 727 applicants. We selected a tree procedure for this analysis because it was easy to present the results to the Yale Medical School admissions committee and because the tree model could serve as a basis for structuring their discussions about admissions policy.

Notice that the values of the predicted variable (the committee's decision to reject or interview) are at the bottom of the tree and the predictors (Medical College Admissions Test and college grade point average) come into the system at each node of the tree.

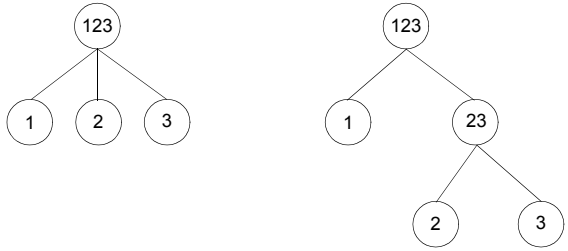
The top node contains the entire sample. Each remaining node contains a subset of the sample in the node directly above it. Furthermore, each node contains the sum of the samples in the nodes connected to and directly below it. The tree thus splits samples.

Each node can be thought of as a cluster of objects, or cases, that is to be split by further branches in the tree. The numbers in parentheses below the terminal nodes show how many cases are incorrectly classified by the tree. A similar tree data structure is used for representing the results of single and complete linkage and other forms of hierarchical cluster analysis (Hartigan, 1975). Tree prediction models add two ingredients: the predictor and predicted variables labeling the nodes and branches.



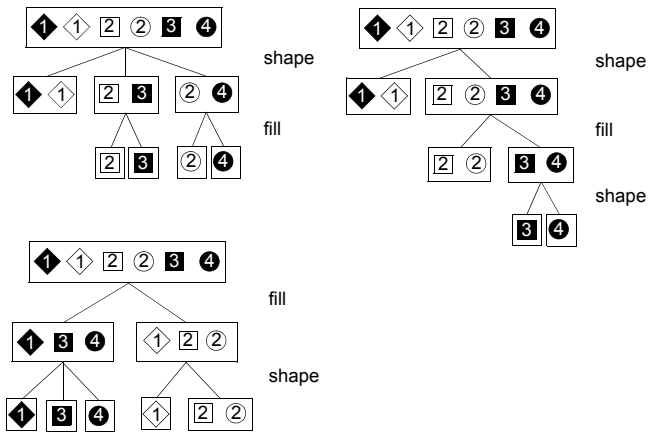
The tree is binary because each node is split into only two subsamples. Classification or regression trees do not have to be binary, but most are. Despite the marketing claims of some vendors, nonbinary, or multibranch, trees are not superior to binary trees. Each is a permutation of the other, as shown in the figure below.

The tree on the left (ternary) is not more parsimonious than that on the right (binary). Both trees have the same number of parameters, or split points, and any statistics associated with the tree on the left can be converted trivially to fit the one on the right. A computer program for scoring either tree (IF ... THEN ... ELSE) would look identical. For display purposes, it is often convenient to collapse binary trees into multibranch trees, but this is not necessary.



Some programs that do multibranch splits do not allow further splitting on a predictor once it has been used. This has an appealing simplicity. However, it can lead to unparsimonious trees. It is unnecessary to make this restriction before fitting a tree.

The figure below shows an example of this problem. The upper right tree classifies objects on an attribute by splitting once on shape, once on fill, and again on shape. This allows the algorithm to separate the objects into only four terminal nodes having common values. The upper left tree splits on shape and then only on fill. By not allowing any other splits on shape, the tree requires five terminal nodes to classify correctly. This problem cannot be solved by splitting first on fill, as the lower left tree shows. In general, restricting splits to only one branch for each predictor results in more terminal nodes.



Categorical or Quantitative Predictors

The predictor variables in the figure on page 43 are quantitative, so splits are created by determining cut points on a scale. If predictor variables are categorical, as in the figure above, splits are made between categorical values. It is not necessary to categorize predictors before computing trees. This is as dubious a practice as recoding data well-suited for regression into categories in order to use chi-square tests. Those who recommend this practice are turning silk purses into sows' ears. In fact, if variables are categorized before doing tree computations, then poorer fits are likely to result. Algorithms are available for mixed quantitative and categorical predictors, analogous to analysis of covariance.

Regression Trees

Morgan and Sonquist (1963) proposed a simple method for fitting trees to predict a quantitative variable. They called the method **Automatic Interaction Detection (AID)**. The algorithm performs stepwise splitting. It begins with a single cluster of cases and searches a candidate set of predictor variables for a way to split the cluster into two clusters. Each predictor is tested for splitting as follows: sort all the n cases on the predictor and examine all $n - 1$ ways to split the cluster in two. For each possible split, compute the within-cluster sum of squares about the mean of the cluster on the dependent variable. Choose the best of the $n - 1$ splits to represent the predictor's contribution. Now do this for every other predictor. For the actual split, choose the predictor and its cut point that yields the smallest overall within-cluster sum of squares.

Categorical predictors require a different approach. Since categories are unordered, all possible splits between categories must be considered. For deciding on one split of k categories into two groups, this means that $2^{k-1}-1$ possible splits must be considered. Once a split is found, its suitability is measured on the same within-cluster sum of squares as for a quantitative predictor.

Morgan and Sonquist called their algorithm AID because it naturally incorporates interaction among predictors. Interaction is not correlation. It has to do, instead, with conditional discrepancies. In the analysis of variance, interaction means that a trend within one level of a variable is not parallel to a trend within another level of the same variable. In the ANOVA model, interaction is represented by cross-products between predictors. In the tree model, it is represented by branches from the same node that have different splitting predictors further down the tree.

The figure below shows a tree without interactions on the left and with interactions on the right. Because interaction trees are a natural by-product of the AID splitting algorithm, Morgan and Sonquist called the procedure "automatic." In fact, AID trees without interactions are quite rare for real data, so the procedure is indeed automatic. To search for interactions using stepwise regression or ANOVA linear modeling, we would have to generate $2^p - p - 1$ interactions among p predictors and compute partial correlations for every one of them in order to decide which ones to include in our formal model.

Classification Trees

Regression trees are parallel to regression/ANOVA modeling, in which the dependent variable is quantitative. Classification trees are parallel to discriminant analysis and algebraic classification methods. Kass (1980) proposed a modification to AID called CHAID for categorized dependent and independent variables. His algorithm incorporated a sequential merge-and-split procedure based on a chi-square test statistic. Kass was concerned about computation time (although this has since proved an unnecessary worry), so he decided to settle for a suboptimal split on each predictor instead of searching for all possible combinations of the categories. Kass's algorithm is like sequential crosstabulation. For each predictor:

- Crosstabulate the m categories of the predictor with the k categories of the dependent variable.
- Find the pair of categories of the predictor whose $2 \times k$ subtable is least significantly different on a chi-square test and merge these two categories.
- If the chi-square test statistic is not "significant" according to a preset critical value, repeat this merging process for the selected predictor until no nonsignificant chi-square is found for a subtable.

- Choose the predictor variable whose chi-square is the largest and split the sample into $l \leq m$ subsets, where l is the number of categories resulting from the merging process on that predictor.
- Continue splitting, as with AID, until no significant chi-squares result.

The CHAID algorithm saves computer time, but it is not guaranteed to find the splits that predict best at a given step. Only by searching all possible category subsets can we do that. CHAID is also limited to categorical predictors, so it cannot be used for quantitative or mixed categorical-quantitative models, as in the figure on page 43. Nevertheless, it is an effective way to search heuristically through rather large tables quickly.

Note: Within the computer science community, there is a categorical splitting literature that often does not cite the statistical work and is, in turn, not frequently cited by statisticians (although this has changed in recent years). Quinlan (1986, 1992), the best known of these researchers, developed a set of algorithms based on information theory. These methods, called ID3, iteratively build decision trees based on training samples of attributes.

Stopping Rules, Pruning, and Cross-Validation

AID, CHAID, and other forward-sequential tree-fitting methods share a problem with other tree-clustering methods—where do we stop? If we keep splitting, a tree will end up with only one case, or object, at each terminal node. We need a method for producing a smaller tree other than the exhaustive one. One way is to use stepwise statistical tests, as in the *F*-to-enter or alpha-to-enter rule for forward stepwise regression. We compute a test statistic (chi-square, *F*, etc.), choose a critical level for the test (sometimes modifying it with the Bonferroni inequality), and stop splitting any branch that fails to meet the test (Wilkinson, 1979, for a review of this procedure in forward selection regression).

Breiman et al. (1984) showed that this method tends to yield trees with too many branches and can also fail to pursue branches that can add significantly to the overall fit. They advocate, instead, pruning the tree. After computing an exhaustive tree, their program eliminates nodes that do not contribute to the overall prediction. They add another essential ingredient, however—the cost of complexity. This measure is similar to other cost statistics, such as Mallows' C_p (Neter et al.,1996), which add a penalty for increasing the number of parameters in a model. Breiman's method is *not* like backward elimination stepwise regression. It resembles forward stepwise regression

with a cutting back on the final number of steps using a different criterion than the *F*-to-enter. This method still cannot do as well as an exhaustive search, which would be prohibitive for most practical problems.

Regardless of how a tree is pruned, it is important to cross-validate it. As with stepwise regression, the prediction error for a tree applied to a new sample can be considerably higher than for the training sample on which it was constructed. Whenever possible, data should be reserved for cross-validation.

Loss Functions

Different loss functions are appropriate for different forms of data. TREES offers a variety of functions that are scaled as proportional reduction in error (PRE) statistics. This allows you to try different loss functions on a problem and compare their predictive validity.

For regression trees, the most appropriate loss functions are least-squares, trimmed mean, and least absolute deviations. Least-squares loss yields the classic AID tree. At each split, cases are classified so that the within-group sum of squares about the mean of the group is as small as possible. The trimmed mean loss works the same way but first trims 20% of outlying cases (10% at each extreme) in a splittable subset before computing the mean and sum of squares. It can be useful when you expect outliers in subgroups and don't want them to influence the split decisions. LAD loss computes least absolute deviations about the mean rather than squares. It, too, gives less weight to extreme cases in each potential group.

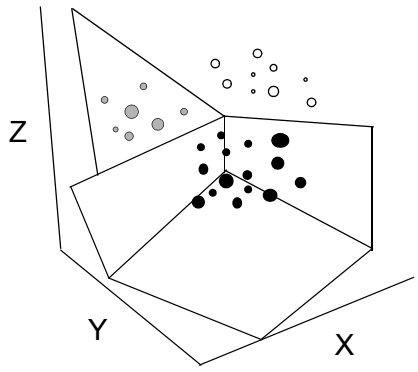
For classification trees, use the phi coefficient (the default), Gini index, or "twoing." The phi coefficient is c^2/n for a $2 \times k$ table formed by the split on k categories of the dependent variable. The Gini index is a variance estimate based on all comparisons of possible pairs of values in a subgroup. Finally, **twoing** is a word coined by Breiman et al. (1984) to describe splitting k categories as if it were a two-category splitting problem. For more information about the effects of Gini and twoing on computations, see Breiman et al. (1984).

Geometry

Most discussions of trees versus other classifiers compare tree graphs and algebraic equations. However, there is another graphic view of what a tree classifier performs. If we look at the cases embedded in the space of the predictor variables, we can ask how

a linear discriminant analysis partitions the cases and how a tree classifier partitions them.

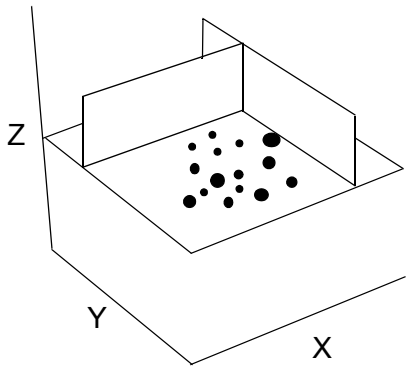
The figure below shows how cases are split by a linear discriminant analysis. There are three subgroups of cases in this example. The cutting planes are positioned approximately halfway between each pair of group centroids. Their orientation is determined by the discriminant analysis. With three predictors and four groups, there are six cutting planes, although only four planes show in the figure. The fourth group is assumed to be under the bottom plane in the figure. In general, if there are *g* groups, the linear discriminant model cuts them with *g(g-1)/2* planes.



The figure below shows how a tree-fitting algorithm cuts the same data. Only the nearest subgroup (dark spots) shows; the other three groups are hidden behind the rear and bottom cutting planes. Notice that the cutting planes are parallel to the axes. While this would seem to restrict the discrimination compared to the more flexible angles allowed by the discriminant planes, the tree model allows interactions between variables, which do not appear in the ordinary linear discriminant model. Notice, for example, that one plane splits on the *X* variable, but the second plane that splits on the *Y* variable cuts only the values to the left of the *X* partition. The tree model can continue to cut any of these subregions separately, unlike the discriminant model, which can cut only globally and with *g(g-1)/2* planes. This is a mixed blessing, however, since tree methods, as we have seen, can over-fit the data. It is critical to test them on new samples.

Tree models are not usually related by authors to dimensional plots in this way, but it is helpful to see that they have a geometric interpretation. Alternatively, we can construct algebraic expressions for trees. They would require dummy variables for any

categorical predictors and interaction (or product) terms for every split whose descendants (or lower nodes) did not involve the same variables on both sides.

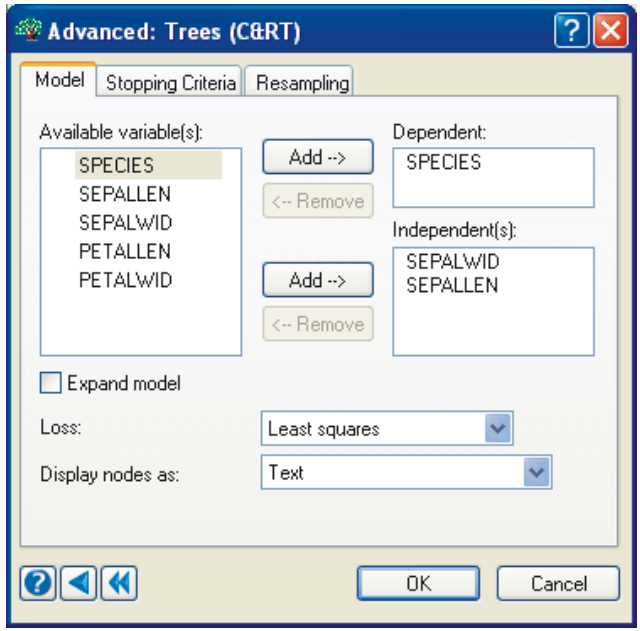


Classification and Regression Trees in SYSTAT

Classification and Regression Trees Dialog Box

To open the Classification and Regression Trees dialog box, from the menus choose:

Advanced
  Trees (C&RT)…



Model selection and estimation are available in the Model tab of the Classification and Regression Trees dialog box:

Dependent. The variable you want to examine. The dependent variable should be continuous or categorical numeric variables (for example, *INCOME*).

Independent(s). Select one or more continuous or categorical variables (grouping variables).

Expand model. Adds all possible sums and differences of the predictors to the model.

Loss. Select a loss function from the drop-down list.

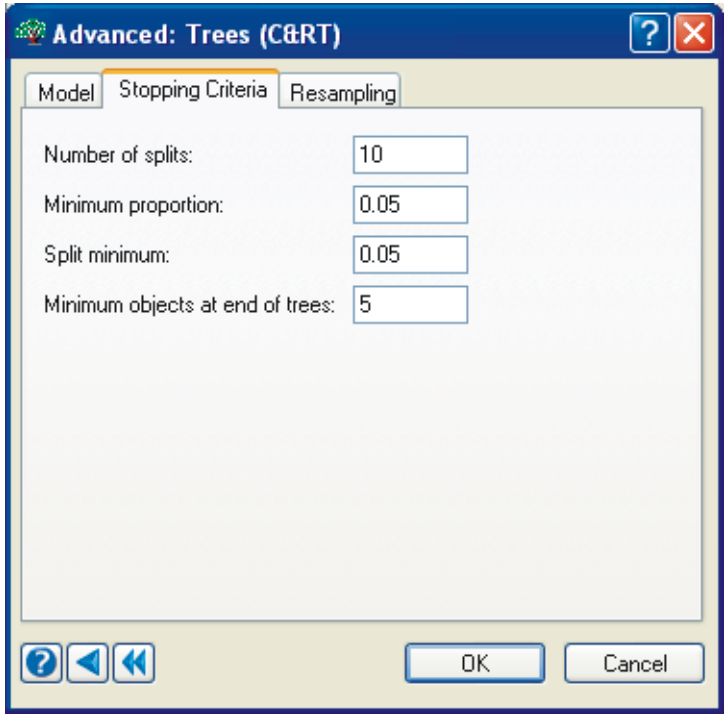
- **Least-squares**. The least-squared loss (AID) minimizes the sum of the squared deviations.
- **Trimmed mean**. The trimmed mean loss (TRIM) "trims" the extreme observations (20%) prior to computing the mean.
- **Least absolute deviations**. The least absolute deviations loss (LAD).
- **Phi coefficient**. The phi coefficient loss computes the correlation between two dichotomous variables.
- **Gini index**. The Gini index loss measures inequality or dispersion.
- **Twoing**. The twoing loss function.

Display nodes as. Select the type of density display. The following types are available:

- **Box plot.** Plot that uses boxes to show a distribution shape, central tendency, and variability.
- **Dot histogram (Dit).** Produces a density display that looks similar to a histogram. Unlike histograms, dot histograms represent every observation with a unique symbol, so they are especially suited for small- to moderate-size samples of continuous data.
- **Symmetrical dot density (Dot).** Plot that displays dots at the exact locations of data values.
- **Jittered dot density.** Plot that calculates the exact locations of the data values, but jitters points randomly on a short vertical axis to keep points from colliding.
- **Density stripes.** Places vertical lines at the location of data values along a horizontal data scale and looks like supermarket bar codes.
- **Text.** Displays text output in the tree diagram including the mode, sample size, and impurity value.

Stopping Criteria

The Stopping Criteria tab contains the parameters for controlling stopping.



Specify the criteria for splitting to stop.

Number of splits. Maximum number of splits.

Minimum proportion. Minimum proportion reduction in error (PRE) for the tree allowed at any split.

Split minimum. Minimum split value allowed at any node.

Minimum objects at end of trees. Minimum count allowed at any node.

Using Commands

After selecting a file with USE FILENAME, continue with:

```
TREES
   MODEL yvar = xvarlist / EXPAND
   ESTIMATE / PMIN=d, SMIN=d, NMIN=n, NSPLIT=n, LOSS=LSQ
TRIM LAD PHI GINI TWOING, DENSITY=STRIPE JITTER DOT DIT BOX
SAMPLE =BOOT(m,n) or SIMPLE(m,n) or JACK
```

Usage Considerations

Types of data. TREES uses rectangular data only.

Print options. The default output includes the splitting history and summary statistics. PLENGTH LONG adds a SYSTAT program for classifying new observations. You can cut and paste this SYSTAT program into a commandspace and submit it to classify new data on the same variables for cross-validation and prediction.

Quick Graphs. TREES produces a Quick Graph for the fitted tree. The nodes may contain text describing split parameters or they may contain density graphs of the data being split. A dashed line indicates that the split is not significant.

Saving files. TREES does not save files. Use the SYSTAT program under PLENGTH LONG to classify your data, compute residuals, etc., on old or new data.

BY groups. TREES analyzes data by groups. Your file need not be sorted on the BY variable(s).

Case frequencies. FREQ <variable> increases the number of cases by the FREQ variable.

Case weights. WEIGHT is not available in TREES.

Examples

The following examples illustrate the features of the TREES module. The first example shows a classification tree for the Fisher-Anderson iris data set. The second example is a regression tree on an example taken from Breiman et al. (1984), and the third is a regression tree predicting the danger of a mammal being eaten by predators.

Example 1
Classification Tree

This example shows a classification tree analysis of the Fisher-Anderson iris data set featured in Discriminant Analysis. We use the Gini loss function and display a graphical tree, or mobile, with dot histograms, or dit plots.

The input is:

```
USE IRIS
LAB SPECIES/1='SETOSA',2='VERSICOLOR',3='VIRGINICA'
TREES
    MODEL SPECIES=SEPALLEN,SEPALWID,PETALLEN,PETALWID
    ESTIMATE/LOSS=GINI,DENSITY=DIT
```

The output is:

```
Categorical Values Encountered during Processing are

 Variables    Levels
-----------------------------------------------
 SPECIES      setosa   versicolor   virginica
 (3levels)

 Split   Variable     PRE    Improvement
-----------------------------------------
 1       PETALLEN   0.500         0.500
 2       PETALWID   0.890         0.390

Fitting Method                        : Gini Index
Predicted Variable                    : SPECIES
Minimum Split Index Value             : 0.050
Minimum Improvement in PRE            : 0.050
Maximum Number of Nodes Allowed       : 21
Minimum Count Allowed in Each Node    : 5
Number of Terminal Nodes in Final Tree : 3
Proportional Reduction in Error (PRE)  : 0.890

 Node From     Count        Mode   Impurity   Split Variable   Cut Value    Fit
-------------------------------------------------------------------------------
 1       0       150                0.333         PETALLEN       3.000  0.500
 2       1        50      setosa    0.000
 3       1       100                0.250         PETALWID       1.800  0.779
 4       3        54  versicolor    0.084
 5       3        46   virginica    0.021
```

The *PRE* for the whole tree is 0.89 (similar to R^2 for a regression model), which is not bad. Before exulting, however, we should keep in mind that while Fisher chose the iris data set to demonstrate his discriminant model on real data, it is barely worthy of the effort. We can classify the data almost perfectly by looking at a scatterplot of petal length against petal width.

The unique SYSTAT display of the tree is called a mobile (Wilkinson, 1995). The dit plots are ideal for illustrating how it works. Imagine each case is a marble in a box

at each node. The mobile simply balances all of the boxes. The reason for doing this is that we can easily see splits that cut only a few cases out of a group. These nodes will hang out conspicuously. It is fairly evident in the first split, for example, which cuts the population into half as many cases on the right (petal length less than 3) as on the left.

This display has a second important characteristic that is different from other tree displays. The mobile coordinates the polarity of the terminal nodes (red on color displays) rather than the direction of the splits. This design has three consequences: we can evaluate the distributions of the subgroups on a common scale, we can see the direction of the splits on each splitting variable, and we can look at the distributions on the terminal nodes from left to right to see how the whole sample is split on the dependent variable.

The first consequence means that every box containing data is a miniature density display of the subgroup's values on a common scale (same limits and same direction). We don't need to "drill down" on the data in a subgroup to see its distribution. It is immediately apparent in the tree. Suppose you prefer box plots or other density displays, simply use:

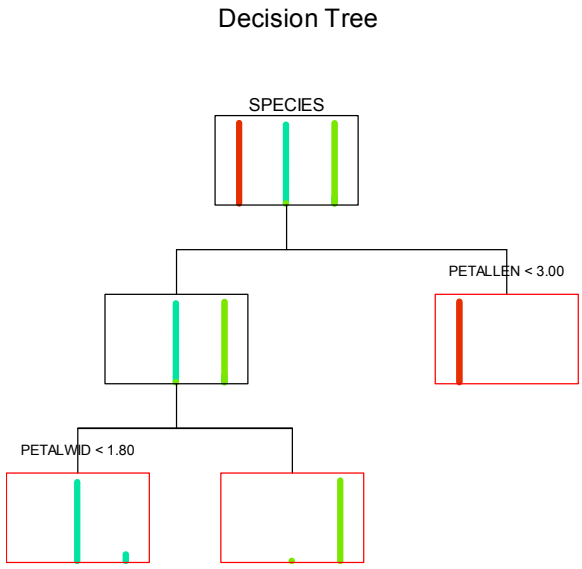
```
DENSITY = BOX
```

or another density as an ESTIMATE option. Dit plots are most suitable for classification trees, however; because they spike at the category values, they look like bar charts for categorical data. For continuous data, dit plots look like histograms. Although they are my favorite density display for this purpose, they can be time consuming to draw on large samples, so text summary is the default graphical display.

The second consequence of ordering the splits according to the polarity of the dependent (rather than the independent) variable is that the direction of the split can be recognized immediately by looking at which side (left or right) the split is displayed on. Notice that *PETALLEN* < 3.000 occurs on the left side of the first split. This means that the relation between petal length and species (coded 1..3) is positive. The same is true for petal width within the second split group because the split banner occurs on the left. Banners on the right side of a split indicate a negative relationship between the dependent variable and the splitting variable within the group being split, as in the regression tree examples.

The third consequence of ordering the splits is that we can look at the terminal nodes from left to right and see the consequences of the split in order. In the present example, notice that the three species are ordered from left to right in the same order that they are coded. You can change this ordering for a categorical variable with the CATEGORY

and ORDER commands. Adding labels, as we did here, makes the output more interpretable.



Example 2
Regression Tree with Box Plots

This example shows a simple AID model. The data set is Boston housing prices, cited in Belsley et al. (1980) and used in Breiman et al. (1984). We are predicting median home values (*MEDV*) from a set of demographic variables.

The input is:

```
USE BOSTON
TREES
      MODEL MEDV=CRIM..LSTAT
      ESTIMATE/PMIN=.005,DENSITY=BOX
```

The output is:

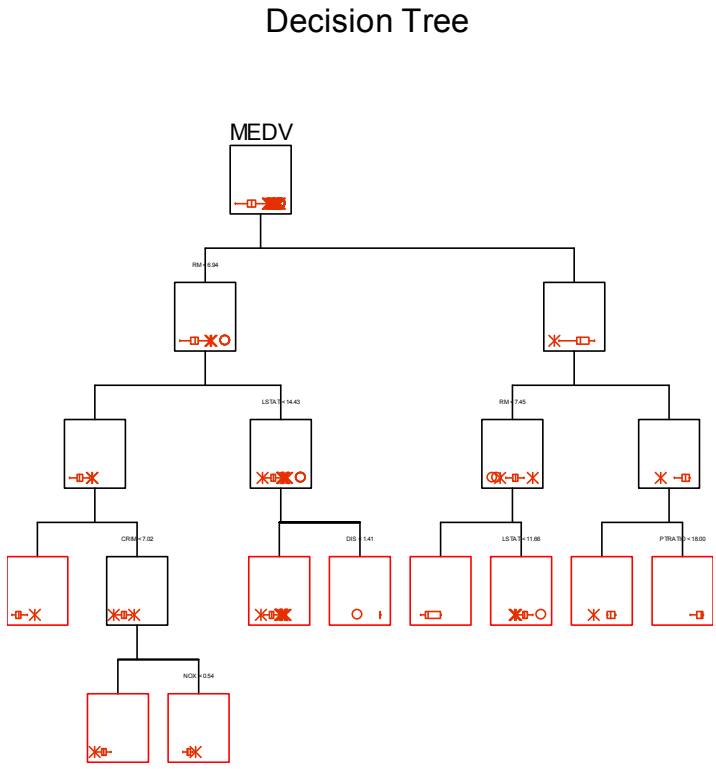
Split	Variable	PRE	Improvement
1	RM	0.453	0.453
2	RM	0.524	0.072
3	LSTAT	0.696	0.171
4	PTRATIO	0.706	0.010
5	LSTAT	0.723	0.017
6	DIS	0.782	0.059
7	CRIM	0.809	0.027
8	NOX	0.815	0.006

```
Fitting Method                          : Least Squares
Predicted Variable                      : MEDV
Minimum Split Index Value               : 0.050
Minimum Improvement in PRE              : 0.005
Maximum Number of Nodes Allowed         : 21
Minimum Count Allowed in Each Node      : 5
Number of Terminal Nodes in Final Tree : 9
Proportional Reduction in Error (PRE)  : 0.815
```

Node	From	Count	Mean	Standard Deviation	Split Variable	Cut Value	Fit
1	0	506	22.533	9.197	RM	6.943	0.453
2	1	430	19.934	6.353	LSTAT	14.430	0.422
3	1	76	37.238	8.988	RM	7.454	0.505
4	3	46	32.113	6.497	LSTAT	11.660	0.382
5	3	30	45.097	6.156	PTRATIO	18.000	0.405
6	2	255	23.350	5.110	DIS	1.413	0.380
7	2	175	14.956	4.403	CRIM	7.023	0.337
8	5	25	46.820	3.768			
9	5	5	36.480	8.841			
10	4	41	33.500	4.594			
11	4	5	20.740	9.080			
12	6	5	45.580	9.883			
13	6	250	22.905	3.866			
14	7	101	17.138	3.392	NOX	0.538	0.227
15	7	74	11.978	3.857			
16	14	24	20.021	3.067			
17	14	77	16.239	2.975			

The Quick Graph of the tree more clearly reveals the sample-size feature of the mobile display. Notice that a number of the splits, because they separate out a few cases only, are extremely unbalanced. This can be interpreted in two ways, depending on context. On the one hand, it can mean that outliers are being separated so that subsequent splits can be more powerful. On the other hand, it can mean that a split is wasted by focusing on the outliers when further splits don't help to improve the prediction. The former case appears to apply in our example. The first split separates out a few expensive housing tracts (the median values have a positively skewed distribution for all tracts), which makes subsequent splits more effective. The box plots in the terminal nodes are

narrow.



Example 3
Regression Tree with Dit Plots

This example involves predicting the danger of a mammal being eaten by predators (Allison and Cicchetti, 1976). The predictors are hours of dreaming and non-dreaming sleep, gestational age, body weight, and brain weight. Although the danger index has only five values, we are treating it as a quantitative variable with meaningful numerical values.

The input is:

```
USE SLEEP
TREES
     MODEL DANGER=BODY_WT,BRAIN_WT,
                  SLO_SLEEP,DREAM_SLEEP,GESTATE
     ESTIMATE / DENSITY=DIT
```

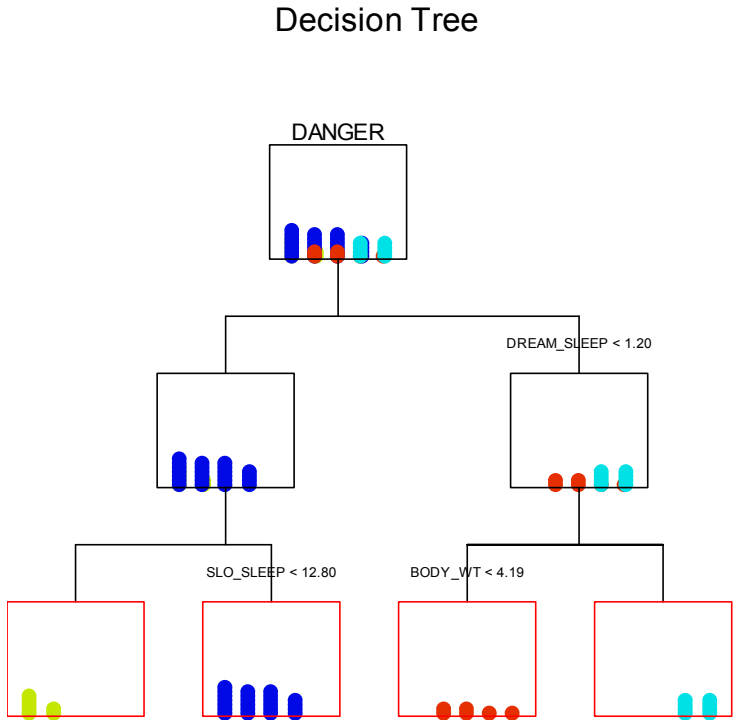
The output is:

```
18 Cases Deleted due to Missing Data.

  Split    Variable     PRE   Improvement
 ----------------------------------------
  1     DREAM_SLEEP   0.404         0.404
  2         BODY_WT   0.479         0.074
  3       SLO_SLEEP   0.547         0.068

Fitting Method                        : Least Squares
Predicted Variable                    : DANGER
Minimum Split Index Value             : 0.050
Minimum Improvement in PRE            : 0.050
Maximum Number of Nodes Allowed       : 21
Minimum Count Allowed in Each Node    : 5
Number of Terminal Nodes in Final Tree : 4
Proportional Reduction in Error (PRE)  : 0.547

 Node   From   Count    Mean    Standard   Split Variable   Cut Value   Fit
                                Deviation
 ------------------------------------------------------------------------------
  1        0      44   2.659       1.380      DREAM_SLEEP       1.200  0.404
  2        1      14   3.929       1.072          BODY_WT       4.190  0.408
  3        1      30   2.067       1.081        SLO_SLEEP      12.800  0.164
  4        2       6   3.167       1.169
  5        2       8   4.500       0.535
  6        3      23   2.304       1.105
  7        3       7   1.286       0.488
```



The prediction is fairly good (*PRE = 0.547*). The Quick Graph of this tree illustrates another feature of mobiles. The dots in each terminal node are assigned a separate color. This way, we can follow their path up the tree each time they are merged. If the prediction is perfect, the top density plot will have colored dots perfectly separated. The extent to which the colors are mixed in the top plot is a visual indication of the badness-of-fit of the model. The fairly good separation of colors for the sleep data is quite clear on the computer screen or with color printing but less evident in a black and white figure.

Computation

Algorithms

TREES uses algorithms from Breiman et al. (1984) for its splitting computations.

Missing Data

Missing data are eliminated from the calculation of the loss function for each split separately.

References

Allison, T. and Cicchetti, D. (1976). Sleep in mammals: Ecological and constitutional correlates. *Science,* 194, 732–734.

Belsley, D. A., Kuh, E., and Welesh, R.E. (1980). *Regression diagnostics: Identifying influential data and sources of collinearity*. New York: John Wiley & Sons.

Bishop, Y. M., Fienberg, S.E., and Holland, P.W. (1975). *Discrete multivariate analysis*. Cambridge, Mass.: MIT Press.

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.I. (1984). *Classification and regression trees*. Belmont, Calif.: Wadsworth.

Einhorn, H. (1972). Alchemy in the behavioral sciences. *Public Opinion Quarterly,* 3, 367–378.

Hartigan, J. A. (1975). *Clustering algorithms*. New York: John Wiley & Sons.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics,* 29, 119–127.

Levine, M. (1991). Statistical analysis for the executive. *Byte,* 17, 183–184.

Milstein, R. M., Burrow, G.N., Willkinson, L., and Kessen, W. (1975). Prediction of screening decisions in a medical school admission process. *Journal of Medical Education,* 51, 626–633.

Morgan, J. N. and Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association,* 58, 415–434.

Neter, J., Kutner, M. H., Nachtscheim, C. J., and Wasserman,W. (1996). *Applied linear statistical models*. 4th.ed. Irwin: McGraw-Hill.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning,* 1, 81–106.

Quinlan, J. R. (1992). *C4.5: Programs for machine learning*. New York: Morgan Kaufmann.

Simon, B. (1991). Knowledge seeker: Statistics for decision makers. *PC Magazine* (January 29), 50.

Wilkinson, L. (1979). Tests of significance in stepwise regression. *Psychological Bulletin,* 86, 168–174.

Wilkinson, L. (1995). Mobiles. Department of Statistics, Northwestern University, Evanston, Ill.