GUIDE Classification and Regression Trees*
User Manual for Version 21.3

Wei-Yin Loh
Department of Statistics
University of Wisconsin–Madison

March 19, 2016

Contents

*Based upon work partially supported by grants from the U.S. Army Research Office, the National Science Foundation and the National Institutes of Health.

1 Warranty disclaimer

Redistribution and use in binary forms, with or without modification, are permitted provided that the following condition is met:

Redistributions in binary form must reproduce the above copyright notice, this condition and the following disclaimer in the documentation and/or other materials provided with the distribution.

2 Introduction

GUIDE stands for *Generalized, Unbiased, Interaction Detection and Estimation.* It is the only classification and regression tree algorithm with all these features:

1. Unbiased variable selection.
2. Kernel and nearest-neighbor node models for classification trees.
3. Weighted least squares, least median of squares, quantile, Poisson, and relative risk (proportional hazards) regression models.
4. Univariate, multivariate, and longitudinal response variables.
5. Pairwise interaction detection at each node.
6. Linear splits on two variables at a time for classification trees.
7. Categorical variables for splitting only, or for both splitting and fitting (via 0-1 dummy variables), in regression tree models.
8. Ranking and scoring of predictor variables.
9. Tree ensembles (bagging and forests).

Table 1: Comparison of GUIDE, QUEST, CRUISE, CART, and C4.5 classification tree algorithms. Node models: S = simple, K = kernel, L = linear discriminant, N = nearest-neighbor.

	GUIDE	QUEST	CRUISE	CART	C4.5
Unbiased splits	Yes	Yes	Yes	No	No
Splits per node	2	2	≥ 2	2	2
Interaction detection	Yes	No	Yes	No	No
Importance ranking	Yes	No	No	Yes	No
Class priors	Yes	Yes	Yes	Yes	No
Misclassification costs	Yes	Yes	Yes	Yes	No
Linear splits	Yes	Yes	Yes	Yes	No
Categorical splits	Subsets	Subsets	Subsets	Subsets	Atoms
Node models	S, K, N	S	S, L	S	S
Missing values	Special	Imputation	Surrogate	Surrogate	Weights
Tree diagrams	Text and LaTeX			Proprietary	Text
Bagging	Yes	No	No	No	No
Forests	Yes	No	No	No	No

Tables 1 and 2 compare the features of GUIDE with CRUISE (Kim and Loh, 2001, 2003), QUEST (Loh and Shih, 1997), C4.5 (Quinlan, 1993), RPART[1], and M5' (Quinlan, 1992; Witten and Frank, 2000).

The GUIDE algorithm is documented in Loh (2002) for regression trees and Loh (2009) for classification trees. Loh (2008a), Loh (2011) and Loh (2014) review the subject. Advanced features of the algorithm are reported in Chaudhuri and Loh (2002), Loh (2006b), Kim et al. (2007), Loh et al. (2007), and Loh (2008b). For a list of third-party applications of GUIDE, CRUISE, QUEST, and the logistic regression tree algorithm LOTUS (Chan and Loh, 2004; Loh, 2006a), see http://www.stat.wisc.edu/~loh/apps.html

This manual illustrates the use of the program and interpretation of the output.

[1]RPART is an implementation of CART (Breiman et al., 1984) in R. CART is a registered trademark of California Statistical Software, Inc.

Table 2: Comparison of GUIDE, CART and M5' regression tree algorithms

	GUIDE	CART	M5'
Unbiased splits	Yes	No	No
Pairwise interaction detection	Yes	No	No
Importance scores	Yes	Yes	No
Loss functions	Weighted least squares, least median of squares, quantile, Poisson, proportional hazards	Least squares, least absolute deviations	Least squares only
Survival, longitudinal and multi-response data	Yes, yes, yes	No, no, no	No, no, no
Node models	Constant, multiple, stepwise linear, polynomial, ANCOVA	Constant only	Constant and stepwise
Linear models	Multiple or stepwise (forward and forward-backward)	N/A	Stepwise
Variable roles	Split only, fit only, both, neither, weight, censored, offset	Split only	Split and fit
Categorical variable splits	Subsets of categorical values	Subsets	0-1 variables
Tree selection	Pruning or stopping rules	Pruning only	Pruning only
Tree diagrams	Text and LaTeX	Proprietary	PostScript
Operation modes	Interactive and batch	Interactive and batch	Interactive
Case weights	Yes	Yes	No
Transformations	Powers and products	No	No
Missing values in split variables	Missing values treated as a special category	Surrogate splits	Imputation
Missing values in linear predictors	Choice of separate constant models or mean imputation	N/A	Imputation
Bagging & forests	Yes & yes	No & no	No & no
Data conversions	ARFF, C4.5, Minitab, R, SAS, Statistica, Systat, CSV	No	No

2.1 Installation

GUIDE is available free from www.stat.wisc.edu/~loh/guide.html in the form of compiled 32- and 64-bit executables for Linux, Mac OS X, and Windows on Intel and compatible processors.

Mac OS X and Linux: Make the unzipped file executable by issuing this command in a `Terminal` application in the folder where the file is located: `chmod a+x guide`

Mac OS X only: The Mac OS X version requires **Xcode** and **gfortran** to be installed. To ensure that the gfortran libraries are placed in the right place, follow the steps:

1. Install **Xcode** from https://developer.apple.com/xcode/downloads/.
2. Go to http://hpc.sourceforge.net and download file `gcc-6.0-bin.tar.gz` to your Downloads folder. The direct link to the file is http://prdownloads.sourceforge.net/hpc/gcc-6.0-bin.tar.gz?download
3. Open a **Terminal** window and type (or copy and paste):
   (a) `cd ~/Downloads`
   (b) `gunzip gcc-6.0-bin.tar.gz`
   (c) `sudo tar -xvf gcc-6.0-bin.tar -C /`

2.2 LaTeX

GUIDE uses the public-domain software LaTeX (http://www.ctan.org) to produce tree diagrams. The specific locations are:

Linux: TeX Live http://www.tug.org/texlive/

Mac: MacTeX http://tug.org/mactex/

Windows: proTeXt http://www.tug.org/protext/

After LaTeX is installed, a pdf file of a LaTeX file, called `diagram.tex` say, produced by GUIDE can be obtained by typing these three commands in a terminal window:

1. `latex diagram`
2. `dvips diagram`

3. `ps2pdf diagram.ps`

The first command produces a file called `diagram.dvi` which the second command uses to create a postscript file called `diagram.ps`. The latter can be viewed and printed if a postscript viewer (such as *Preview* for the Mac) is installed. If no postscript viewer is available, the last command can be used to convert the postscript file into a pdf file, which can be viewed and printed with *Adobe Reader*. Windows users who want to insert the diagram in a Microsoft Word or PowerPoint file should convert the postscript file `diagram.ps` to *Windows Meta File* format (`.wmf`) (using Adobe Illustrator or any free graphics conversion program).

The file `diagram.tex` can be edited to change colors, node sizes, etc. See, e.g., http://tug.org/PSTricks/main.cgi/.

3 Program operation

3.1 Required files

The GUIDE program requires two text files for input.

Data file: This file contains the training sample. Each file record consists of observations on the response (i.e., dependent) variable, the predictor (i.e., X or independent) variables, and optional weight and time variables. Entries in each record are comma, space, or tab delimited (multiple spaces are treated as one space, but not for commas). A record can occupy more than one line in the file, but each record must begin on a new line.

Values of categorical variables can contain any ascii character except single and double quotation marks, which are used to enclose values that contain spaces and commas. Values can be up to 60 characters long. Class labels are truncated to 10 characters in tabular displays.

Note: Starting with version 21.0, the data file can contain one or more header lines (e.g., containing variable names). The next paragraph explains how to skip the header lines when reading the data.

Description file: This provides information about the name and location of the data file, names and column positions of the variables, and their roles in the analysis. This file permits different models to be fitted by changing the roles of the variables. We use the files `irisdsc.txt` and `irisdata.txt` (both obtainable from http://www.stat.wisc.edu/~loh/guide.html) to illustrate. The data give the sepal lengths and widths and the petal lengths and widths of 150 iris flowers. The response variable is the type of iris flower. The contents of `irisdesc.txt` are:

```
irisdata.txt
"?"
1
1 sepallen n
2 sepalwid n
3 petallen n
4 petalwid n
5 class d
```

The first line of the file `irisdsc.txt` gives the name of the training sample file. If the data file `irisdata.txt` is not in the folder where GUIDE is installed, its full path (such as `"c:\data\irisdata.txt"`) is needed. The second line gives the the missing value code, which can be up to 80 characters long. If it contains non alphanumeric characters, it must be surrounded by quotation marks. A missing value code must appear in the second line of the file even if there are no missing values in the data (in that case any character string not present among the data values can be used). **The third line gives the line number of the first data record in the data file. In this example, the data file has no header lines. Thus the "1" on the third line of irisdesc.txt indicates that the first data record starts on line 1 of irisdata.txt. Note that blank lines in the data file are ignored, i.e., they are not counted.** The position, name and role of each variable comes next (in that order), with one line for each variable.

Variable names must begin with an alphabet and be not more than 60 characters long. If a name contains non-alphanumeric characters, it must be enclosed in matching single or double quotes. Spaces and the four characters `#`, `%`, {, and } are replaced by dots (periods) if they appear in a name. Variable names are truncated to 10 characters in tabular output. Leading and trailing spaces are dropped.

The following roles for the variables are permitted. Lower and upper case letters are accepted.

b Categorical variable that is used **b**oth for splitting and for node modeling in regression. It is transformed to 0-1 dummy variables for node modeling. It is converted to **c** type for classification.

- **c** **C**ategorical variable used for splitting only.
- **d** **D**ependent variable. Except for multi-response data (see Sec. 5.8), there can only be one such variable. In the case of relative risk models, this is the **d**eath indicator. The variable can take character string values for classification.
- **f** Numerical variable used only for **f**itting the linear models in the nodes of the tree. It is not used for splitting the nodes and is disallowed in classification.
- **n** **N**umerical variable used both for splitting the nodes and for fitting the node models. It is converted to type **s** in classification.
- **r** Categorical treatment (**R**x) variable used only for fitting the linear models in the nodes of the tree. It is not used for splitting the nodes. If this variable is present, all **n** variables are automatically changed to **s**.
- **s** Numerical-valued variable only used for **s**plitting the nodes. It is not used as a regressor in the linear models. This role is suitable for ordinal categorical variables if they are given numerical values that reflect the orderings.
- **t** Survival **t**ime (for proportional hazards models) or observation **t**ime (for longitudinal models) variable.
- **w** **W**eight variable for weighted least squares regression or for excluding observations in the training sample from tree construction. See section 8.2 for the latter. Except for longitudinal models, a record with a missing value in a `d`, `t`, or `z`-variable is automatically assigned zero weight.
- **x** E**x**cluded variable. This allows models to be fitted to different subsets of the variables without reformatting the data file.
- **z** **O**ffset variable used only in Poisson regression.

GUIDE runs within a **terminal window** of the computer operating system.

Do not double-click its icon!

Linux. Any terminal program will do.

Mac OS X. The program is called **Terminal**; it is in the **Applications Folder**.

Windows. The terminal program is started from the **Start button** by choosing **All Programs → Accessories → Command Prompt**

3.2 Input file creation

GUIDE is started by typing its (lowercase) name in a terminal. The preferred way is to create an input file (option 1 below) for subsequent execution. The input file may be edited if you wish to change some input parameters later. In the following, the sign (>) is the terminal prompt (not to be typed!).

```
> guide
GUIDE Classification and Regression Trees and Forests
Compiled with GFortran 6.0 on Mac OS X El Capitan 10.11.1
Version 21.3 (Build date: March 6, 2016)
Copyright (c) 1997-2016 Wei-Yin Loh. All rights reserved.
This software is based upon work supported by the U.S. Army Research Office,
the National Science Foundation and the National Institutes of Health.

Choose one of the following options:
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice:
```

The meanings of these options are:

0. Print the warranty disclaimer.

1. Create an input file for subsequent execution.

2. Run the program right away without creating an input file.

3. Convert the data file into a format suitable for importation into database, spreadsheet, or statistics software. See Table 2 for the statistical packages supported. Section 8.5 has an example.

4. Importance scoring of variables and identification of items with differential item functioning.

4 Classification

4.1 Default: univariate splits

We first show how to generate an input file to produce a classification tree from the data in the file `irisdata.txt`, using the default options. Whenever you are prompted for a selection, there is usually range of permissible values given within square brackets and a default choice (indicated by the symbol `<cr>=`). The default may be selected by pressing the `ENTER` or `RETURN` key. Annotations are printed in *blue italics* in this manual.

4.1.1 Input file creation with default options

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice: 1
Name of batch input file: irisin.txt
 This file will store your answers to the prompts.
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for
data conversion ([1:3], <cr>=1):
 Press the ENTER or RETURN key to accept the default selection.
Name of batch output file: irisout.txt
 This file will contain the results when you apply the input file to GUIDE later.
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
 Option 2 is for bagging and random forest-type methods.
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):
 The default option will produce a traditional classification tree.
 Choose option 2 for more advanced features.
Input name of data description file (maximum 100 characters; enclose within quotes
if it contains spaces or non alphanumeric characters): irisdsc.txt
Reading data description file ...
Training sample file: irisdata.txt
 The name of the data set is read from the description file.
 Some information about the data are printed in the next few lines.
Missing value code: ?
Records in data file start on line 1
Warning: N variables changed to S
 This warning is triggered because we are fitting a classification model.
Dependent variable is class
Length of longest data entry = 11
Total number of cases =         150
Number of classes =           3
Checking data ...
Class name          Num. cases   Proportion
Setosa                      50   0.33333333
Versicolour                 50   0.33333333
Virginica                   50   0.33333333
```

```
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals    #X-var    #N-var    #F-var    #S-var    #B-var    #C-var
       150          0          0         0         0         0         4         0         0
 No. cases used for training =         150
 Choose 1 for estimated priors, 2 for equal priors, 3 to input the priors from a file
 Input 1, 2, or 3 ([1:3], <cr>=1):
  See below for examples of equal priors and specified priors.
 Choose 1 for unit misclassification costs, 2 to input costs from a file
 Input 1 or 2 ([1:2], <cr>=1):
 Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
  Choose option 2 if you do not want LaTeX code.
 Input file name to store LaTeX code (use .tex as suffix): iristree.tex
 Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
 Input name of file to store node ID and fitted value of each case: irisfit.txt
  This file will contain the node number and predicted class for each observation.
 Input file is created!
 Run GUIDE with the command: guide < irisin.txt
 Press ENTER or RETURN to quit
```

4.1.2 Contents of irisin.txt

Here are the contents of the input file:

```
123321       (do not edit this file unless you know what you are doing)
  21.3       (version of GUIDE that generated this file)
 1           (1=model fitting, 2=importance or DIF scoring, 3=data conversion)
"irisout.txt"  (name of output file)
 1           (1=one tree, 2=ensemble)
 1           (1=classification, 2=regression, 3=propensity score grouping)
 1           (1=simple model, 2=nearest-neighbor, 3=kernel)
 1           (0=linear 1st, 1=univariate 1st, 2=skip linear, 3=skip linear and interaction)
 1           (1=prune by CV, 2=by test sample, 3=no pruning)
"irisdsc.txt"  (name of data description file)
        10   (number of cross-validations)
 1           (1=mean-based CV tree, 2=median-based CV tree)
     0.500   (SE number for pruning)
 1           (1=estimated priors, 2=equal priors, 3=other priors)
 1           (1=unit misclassification costs, 2=other)
 2           (1=split point from quantiles, 2=use exhaustive search)
 1           (1=default max number of split levels, 2=specify no. in next line)
 1           (1=default min node size, 2=specify node size in next line)
 1           (1=write latex, 2=skip latex)
"iristree.tex" (latex file name)
 1           (1=vertical tree, 2=sideways tree)
 1           (1=include node numbers, 2=exclude)
```

```
 1          (1=number all nodes, 2=only terminal nodes)
 1          (1=color terminal nodes, 2=no colors)
 1          (0=#errors, 1=class sizes in nodes, 2=nothing)
 1          (1=no storage, 2=store fit and split variables, 3=store split variables and values)
 2          (1=do not save individual fitted values and node IDs, 2=save in a file)
"irisfit.txt" (file name for fitted values and node IDs)
 1          (1=do not save terminal node IDs for importance scoring in a file, 2=save them)
 1          (1=do not write R function, 2=write R function)
```

GUIDE reads only the first entry in each line; the remainder of the line is for human consumption. Because each question depends on the answers you have given to previous questions, only some of the entries in the input file may be changed. Examples are the SE number, maximum number of split levels, minimum node size, vertical vs. sideways tree diagram, coloring terminal nodes, and printing of class sizes in the LaTeX tree diagram.

4.1.3 Executing the program and interpreting the output

Once the input file is generated, GUIDE can be executed with the command:

```
guide < irisin.txt
```

Following is an annotated copy of the contents of the output file.

```
Classification tree
Pruning by cross-validation
Data description file: irisdsc.txt
Training sample file: irisdata.txt
Missing value code: ?
Records in data file start on line 1
 This says that the first record begins on line 1 of the data file.
Warning: N variables changed to S
Dependent variable is class
Number of records in data file: 150
Length of longest data entry: 11
Number of classes =  3
      Class      #Cases    Proportion
Setosa              50    0.33333333
Versicolour         50    0.33333333
Virginica           50    0.33333333
 This gives the number of observations in each class.
Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
```

```
 Column  Name            Minimum      Maximum   #Categories       #Missing
      1  sepallen   s   4.3000E+00   7.9000E+00
      2  sepalwid   s   2.0000E+00   4.4000E+00
      3  petallen   s   1.0000E+00   6.9000E+00
      4  petalwid   s   1.0000E-01   2.5000E+00
      5  class      d                                   3
```

This shows the type and minimum and maximum values of each ordered variable.

```
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       150          0          0        0        0        0        4        0        0
No. cases used for training: 150
```

This shows the number of each type of variable.

```
Univariate split highest priority
Interaction and linear splits 2nd and 3rd priorities
Pruning by v-fold cross-validation, with v =  10
Selected tree is based on mean of CV estimates
Simple node models
Estimated priors
Unit misclassification costs
Split values for N and S variables based on exhaustive search
Max number of split levels =  10
Minimum node size =  3

Pruning sequence
  Subtree     Pruned  #Terminal        True           Geometric
   number       node      nodes        alpha             mean
        0          0          6       0.0000          0.0000
        1         13          5       0.0000          0.0000
        2          6          3      0.33333E-02     0.31269E-01
        3          3          2      0.29333         0.31269
        4          1          1      0.33333         0.17977+309

Number of SE's for pruned tree =   5.0000E-01

Size and CV mean cost and SE of subtrees:
 Tree    #Tnodes  Mean Cost   SE(Mean)   BSE(Mean)  Median Cost  BSE(Median)
   0        6   5.333E-02   1.835E-02   8.222E-03   6.667E-02   5.692E-03
   1**      5   5.333E-02   1.835E-02   8.222E-03   6.667E-02   5.692E-03
   2++      3   7.333E-02   2.128E-02   1.457E-02   6.667E-02   1.859E-02
   3        2   3.333E-01   3.849E-02   0.000E+00   3.333E-01   0.000E+00
   4        1   6.667E-01   3.849E-02   0.000E+00   6.667E-01   0.000E+00

0-SE tree based on mean is marked with *
0-SE tree based on median is marked with +
Selected-SE tree based on mean using naive SE is marked with **
Selected-SE tree based on mean using bootstrap SE is marked with --
```

```
Selected-SE tree based on median and bootstrap SE is marked with ++
** tree same as -- tree
+ tree same as ++ tree
* tree same as ** tree
* tree same as -- tree
 The tree with the smallest mean CV cost is marked with an asterisk.
 The selected tree is marked with two asterisks; it is the smallest one
 having mean CV cost within the specified standard error (SE) bounds.
 The mean CV costs and SEs are given in the 3rd and 4th columns.
 The other columns are bootstrap estimates used for experimental purposes.

Following tree is based on mean CV with naive SE estimate (**).

Structure of final tree. Each terminal node is marked with a T.

Node cost is node misclassification cost divided by number of training cases
       Node    Total    Train    Predicted         Node    Split          Interacting
      label    cases    cases    class             cost    variables      variable
          1      150      150    Setosa          6.667E-01 petalwid
         2T       50       50    Setosa          0.000E+00 -
          3      100      100    Versicolour     5.000E-01 petalwid
          6       54       54    Versicolour     9.259E-02 sepalwid :petalwid
         12       19       19    Versicolour     1.579E-01 petalwid
        24T       14       14    Versicolour     0.000E+00 -
        25T        5        5    Virginica       4.000E-01 -
        13T       35       35    Versicolour     5.714E-02 petallen
         7T       46       46    Virginica       2.174E-02 -
 This shows the tree structure in tabular form.  A node with label k has its left
 and right child nodes are labeled 2k and 2k+1, respectively. Terminal nodes are
 indicated with the symbol T. The notation ``:petawid'' in node 6 indicates that
 the variable petalwid has an interaction with the split variable sepalwid.

Number of terminal nodes of final tree: 5
Total number of nodes of final tree:    9

Classification tree:
 The tree structure is shown next in indented text form.
 Node 1: petalwid <=    0.80000
   Node 2: Setosa
 Node 1: petalwid >    0.80000 or ?
   Node 3: petalwid <=    1.75000 or ?
     Node 6: sepalwid <=    2.60000
       Node 12: petalwid <=    1.35000 or ?
         Node 24: Versicolour
       Node 12: petalwid >    1.35000 and not ?
         Node 25: Virginica
```

```
      Node 6: sepalwid >    2.60000 or ?
        Node 13: Versicolour
    Node 3: petalwid >    1.75000 and not ?
      Node 7: Virginica

 Number of cases used for tree construction =  150
 Number misclassified =  5
 Resubstitution est. of mean misclassification cost =    3.3333333333333333E-002

  This is the mean misclassification cost estimated from the training sample.
 Observed and fitted values are stored in irisfit.txt
 LaTeX code for tree is in iristree.tex
```

The left side of Figure 1 shows the classification tree drawn by LaTeX using the file `iristree.tex` and the top lines of the file `irisfit.txt` are shown below. The order of the lines correspond to the order of the observations in the training sample file. The first column (labeled `train`) indicates whether the observation is used ("`y`") or not used ("`n`") to fit the model. Since we used the entire data set to fit the model here, all the entries in the first column are `y`.

```
  train         node  observed  predicted
      y            2    "Setosa"  "Setosa"
      y            2    "Setosa"  "Setosa"
      y            2    "Setosa"  "Setosa"
      y            2    "Setosa"  "Setosa"
      y            2    "Setosa"  "Setosa"
      y            2    "Setosa"  "Setosa"
      y            2    "Setosa"  "Setosa"
```

4.2 Non-default options

4.2.1 Linear splits

The above example uses the default options for classification trees. Other features are available with non-default options. We show how to obtain the linear splits shown in Figure 1 here.

```
 0. Read the warranty disclaimer
 1. Create an input file for batch run
 2. Fit a model without creating input file
 3. Convert data to other formats
 4. Variable importance scoring and differential item functioning
 Input your choice: 1
 Name of batch input file: input2.txt
```

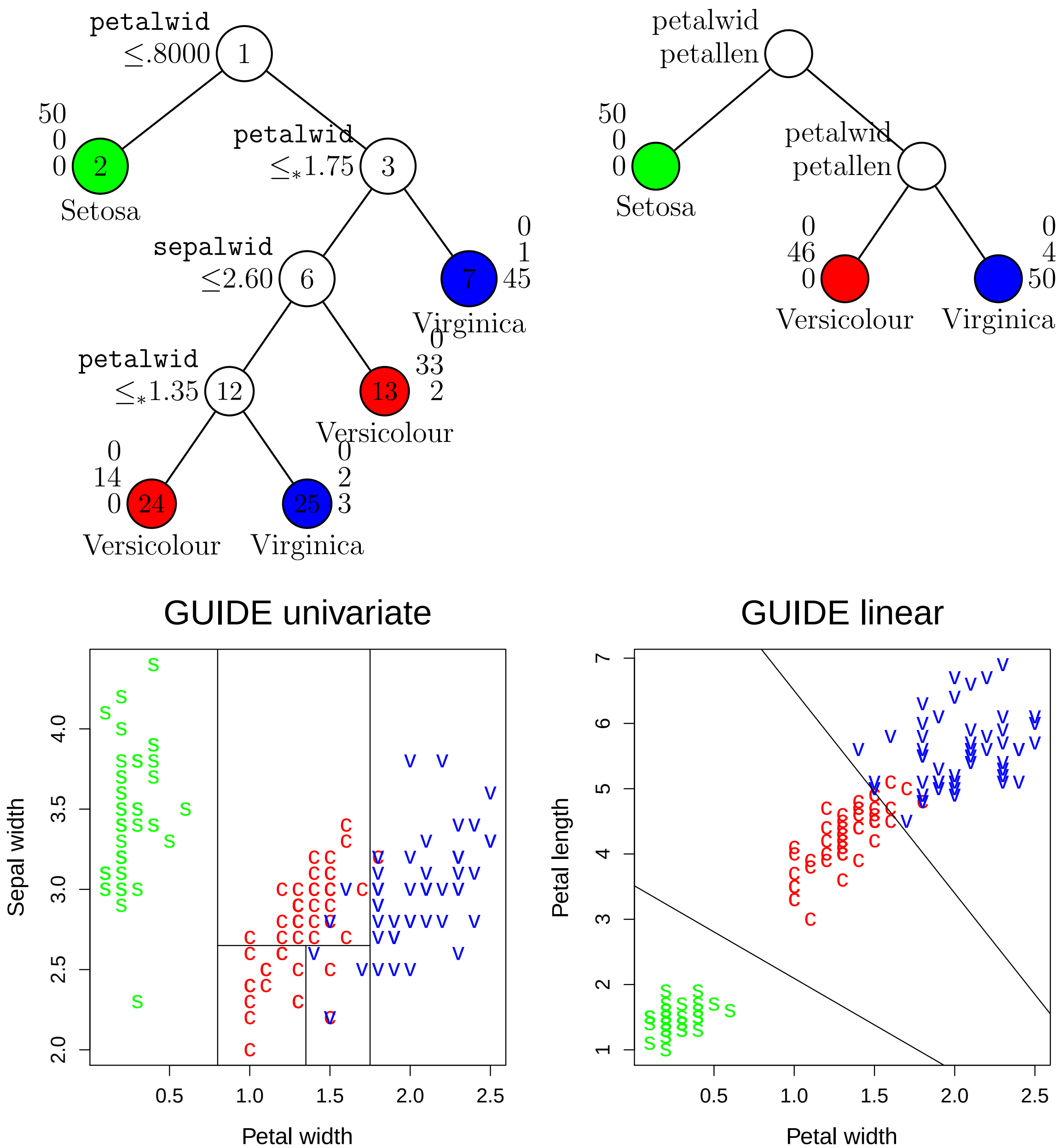

Figure 1: GUIDE classification trees and plots of the data partitions for the iris data using estimated priors and unit misclassification costs. The tree on the left uses univariate splits. At each intermediate node, an observation goes to the left branch if and only if the condition is satisfied. The symbol $\leq_*$ denotes $\leq$ or missing. The tree on the right uses linear splits on two variables at a time. Predicted classes (based on estimated misclassification cost) below terminal nodes; sample sizes for Setosa, Versicolour, and Virginica, respectively, beside nodes.

```
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: output2.txt
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1): 2
 Choosing 2 opens up the other options.
Input 1 for simple, 2 for nearest-neighbor, 3 for kernel method
([1:3], <cr>=1):
 Options 2 and 3 yield nearest-neighbor and kernel discriminant node models.
Input 0 for linear, interaction and univariate splits (in this order),
      1 for univariate, linear and interaction splits (in this order),
      2 to skip linear splits,
      3 to skip linear and interaction splits:
Input your choice ([0:3], <cr>=1): 0
 Option 1 is the default.
Input 1 to prune by CV, 2 by test sample, 3 for no pruning ([1:3], <cr>=1):

Input name of data description file (maximum 100 characters; enclose within quotes
if it contains spaces or non alphanumeric characters): irisdsc.txt
Reading data description file ...
Training sample file: irisdata.txt
Missing value code: ?
Warning: N variables changed to S
Dependent variable is class
Length of longest data entry = 11
Total number of cases =         150
Number of classes =           3
Checking data ...
Class name          Num. cases   Proportion
Setosa                      50   0.33333333
Versicolour                 50   0.33333333
Virginica                   50   0.33333333
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals    #X-var    #N-var    #F-var    #S-var    #B-var    #C-var
       150          0          0         0         0         0         4         0         0
No. cases used for training =         150
Default number of cross-validations =          10
Input 1 to accept the default, 2 to change it ([1:2], <cr>=1):
Best tree may be chosen based on mean or median CV estimate
Input 1 for mean-based, 2 for median-based ([1:2], <cr>=1):
Input number of SEs for pruning ([0.00:1000.00], <cr>=0.50):
Choose 1 for estimated priors, 2 for equal priors, 3 to input the priors from a file
Input 1, 2, or 3 ([1:3], <cr>=1):
Choose 1 for unit misclassification costs, 2 to input costs from a file
Input 1 or 2 ([1:2], <cr>=1):
Choose a split point selection method for numerical variables:
```

```
Choose 1 to use faster method based on sample quantiles
Choose 2 to use exhaustive search
Input 1 or 2 ([1:2], <cr>=2):
Default max number of split levels =          10
Input 1 to accept this value, 2 to change it ([1:2], <cr>=1):
Default minimum node sample size is 10
Input 1 to use the default value, 2 to change it ([1:2], <cr>=1):
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): tree2.tex
Input 1 for a vertical tree, 2 for a sideways tree ([1:2], <cr>=1):
Input 1 to include node numbers, 2 to omit them ([1:2], <cr>=1): 2
 Choosing 2 will give a tree with no node labels.
Input 1 to color terminal nodes, 2 otherwise ([1:2], <cr>=1):
Choose amount of detail in nodes of LaTeX tree diagram
Input 0 for #errors, 1 for class sizes, 2 for nothing ([0:2], <cr>=1):
 Choose 2 for very large trees.
You can store the variables and/or values used to split and fit in a file
Choose 1 to skip this step, 2 to store split variables and their values
Input your choice ([1:2], <cr>=1):
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
Input name of file to store node ID and fitted value of each case: fit2.txt
Input 2 to save terminal node IDs for importance scoring; 1 otherwise ([1:2], <cr>=1):
Input 2 to write R function for predicting new cases, 1 otherwise ([1:2], <cr>=1): 2
Input file name: pred.r
Input file is created!
```

Running the program with this input file yields the following results and the LaTeX tree diagram and partitions on the right side of Figure 1.

```
 Node 1:  7.0205078E-01 * petallen + petalwid <=  2.4700244E+00 or ?
   Node 2: Setosa
 Node 1:  7.0205078E-01 * petallen + petalwid >  2.4700244E+00
   Node 3:  3.2242660E-01 * petallen + petalwid <=  3.0960117E+00 or ?
     Node 6: Versicolour
   Node 3:  3.2242660E-01 * petallen + petalwid >  3.0960117E+00
     Node 7: Virginica
```

The R file `pred.r` contains this function

```
predicted <- function(){
    if(is.na(petalwid) | is.na(petallen) | 0.70205077945722660 * petallen + petalwid
       <= 2.4700244096702053){
        nodeid <- 2
        predict <- "Setosa"
    } else {
        if(is.na(petalwid) | is.na(petallen) | 0.32242659679310148 * petallen + petalwid
           <= 3.0960116541258524){
```

```
            nodeid <- 6
            predict <- "Versicolour"
        } else {
            nodeid <- 7
            predict <- "Virginica"
        }
    }
    return(c(nodeid,predict))
}
```

4.2.2 Equal priors

If a data set has one dominant class, a classification tree will often be null after pruning, because it is hard to beat the classifier that predicts every observation to belong to the dominant class. One way to obtain a non-null tree is to specify equal priors. We illustrate this with the hepatitis data set from http://archive.ics.uci.edu/ml/datasets/Hepatitis. The files hepdsc.txt and hepdat.txt are obtainable from http://www.stat.wisc.edu/~loh/guide.html. The data consist of observations from 155 individuals, of whom 32 are labeled "die" and 123 labeled "live". The contents of the description file hepdsc.txt are:

```
hepdat.txt
"?"
 1
 1  CLASS  d
 2  AGE  n
 3  SEX  c
 4  STEROID  c
 5  ANTIVIRALS  c
 6  FATIGUE  c
 7  MALAISE  c
 8  ANOREXIA  c
 9  BIGLIVER  c
10  FIRMLIVER  c
11  SPLEEN  c
12  SPIDERS  c
13  ASCITES  c
14  VARICES  c
15  BILIRUBIN  n
16  ALKPHOSPHATE  n
17  SGOT  n
18  ALBUMIN  n
19  PROTIME  n
20  HISTOLOGY c
```

Using the default estimated priors yields a null tree with no splits. To obtain a nonnull tree, choose "2" for equal priors in this dialog step:

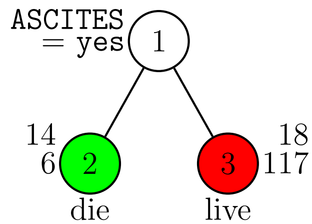

Figure 2: GUIDE 0.50-SE classification tree for predicting CLASS with equal priors or unequal misclassification costs. At each split, an observation goes to the left branch if and only if the condition is satisfied. Predicted classes (based on estimated misclassification cost) printed below terminal nodes; sample sizes for die and live, respectively, beside nodes.

```
Choose 1 for estimated priors, 2 for equal priors, 3 to input the priors from a file
Input 1, 2, or 3 ([1:3], <cr>=1): 2
```

The resulting tree in text form is:

```
Node 1: ASCITES = "yes"
  Node 2: die
Node 1: ASCITES /= "yes"
  Node 3: live
```

The tree drawn by LaTeX is shown in Figure 2. Nodes that predict the same class have the same color. Since the ratio of "die" to "live" classes is 32:123, the effect of equal priors is to treat one "die" observation as equivalent to $r = 123/32 = 3.84375$ "live" observations. Therefore a terminal node is classified as "die" if its ratio of "live" to "die" observations is less than r.

4.2.3 Unequal misclassification costs: hepatitis data

So far, we have assumed that the cost of misclassifying a "die" observation as "live" is the same as the opposite. Another way to obtain a nonnull tree for the hepatitis data is to use unequal misclassification costs. For example, if we think that the cost of misclassifying a "die" observation as "live" is four times that of the opposite, we will use the misclassification cost matrix

$$C = \begin{pmatrix} 0 & 1 \\ 4 & 0 \end{pmatrix}$$

where $C(i, j)$ denotes the cost of classifying an observation as class i given that it belongs to class j. Note that GUIDE sorts the class values in alphabetical order, so that "die" is treated as class 1 and "live" as class 2 here. This matrix is saved in the text file `cost.txt` which has these two lines:

```
0 1
4 0
```

The following lines in the input file generation step shows where this file is used:

```
Choose 1 for estimated priors, 2 for equal priors, 3 to input the priors from a file
Input 1, 2, or 3 ([1:3], <cr>=1):
Choose 1 for unit misclassification costs, 2 to input costs from a file
Input 1 or 2 ([1:2], <cr>=1): 2
Input the name of a file containing the cost matrix C(i|j),
where C(i|j) is the cost of classifying class j as class i
The rows of the matrix must be in alphabetical order of the class names
Input name of file: cost.txt
```

The resulting tree is the same as that in Figure 2.

4.2.4 Nearest-neighbor estimates: car data

The data file `drive.txt` gives the specifications and prices of 428 new cars in the 2004 model year. The data come from the *J. Statistics Education* website http://www.amstat.org/publications/jse/jse_data_archive.htm. Using the description file `drivedsc.txt` whose contents follow, the tree model for predicting `Drive` type misclassifies 90 of the 428 observations.

```
drive.txt
"*"
1
1  Region x
2  Import x
3  Make c
4  Model x
5  Type c
6  Drive d
7  SC x
8  SUV x
9  Wagon x
10 Minivan x
11 Pickup x
12 Allwheel x
13 Rearwheel x
14 Rprice n
15 Dcost n
16 Enginsz n
```

```
17 Cylin n
18 Hp n
19 City n
20 Hwy n
21 Weight n
22 Whlbase n
23 Length n
24 Width n
```

In the examples so far, the observations in each terminal node of a classification tree are all predicted to belong to the class that minimizes the node misclassification cost. This method can be inefficient if the data are difficult to classify with a small number of splits. Alternatively, we can fit a classification model to the data in each node and use it to classify individual observations in the node. GUIDE has two means to achieve this: nearest-neighbor and kernel discrimination. For nearest-neighbor, an observation in a node is classified to the plurality class among observations within its neighborhood. The neighborhood is defined to be the whole node if the split variable is categorical. We illustrate this for the car data with the following input file generation log.

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice: 1
Name of batch input file: drive.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: drive.out
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1): 2
Input 1 for simple, 2 for nearest-neighbor, 3 for kernel method ([1:3], <cr>=1): 2
 Choose nearest-neighbor method here.
Input 1 for univariate, 2 for bivariate preference ([1:2], <cr>=1):
 Default is univariate kernels.
Input 1 for interaction tests, 2 to skip them ([1:2], <cr>=1):
Input 1 to prune by CV, 2 by test sample, 3 for no pruning ([1:3], <cr>=1):

Input name of data description file (maximum 100 characters; enclose within quotes
if it contains spaces or non alphanumeric characters): drivedsc.txt
Reading data description file ...
Training sample file: drive.txt
Missing value code: *
Records in data file start on line 1
Warning: N variables changed to S
```

```
Dependent variable is Drive
Length of longest data entry = 26
Total number of cases =        428
Number of classes =           3
Col. no. Categorical variable    #levels    #missing values
       3 Make                         38                  0
       5 Type                          6                  0
Checking data ...
Class name          Num. cases   Proportion
4wd                         94   0.21962617
fwd                        224   0.52336449
rwd                        110   0.25700935
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       428          0          0       10        0        0       11        0        2
No. cases used for training =         428
Default number of cross-validations =          10
Input 1 to accept the default, 2 to change it ([1:2], <cr>=1):
Best tree may be chosen based on mean or median CV estimate
Input 1 for mean-based, 2 for median-based ([1:2], <cr>=1):
Input number of SEs for pruning ([0.00:1000.00], <cr>=0.50):
Choose 1 for estimated priors, 2 for equal priors, 3 to input the priors from a file
Input 1, 2, or 3 ([1:3], <cr>=1):
Choose 1 for unit misclassification costs, 2 to input costs from a file
Input 1 or 2 ([1:2], <cr>=1):
Choose a split point selection method for numerical variables:
Choose 1 to use faster method based on sample quantiles
Choose 2 to use exhaustive search
Input 1 or 2 ([1:2], <cr>=2):
Default max number of split levels =          10
Input 1 to accept this value, 2 to change it ([1:2], <cr>=1):
Default minimum node sample size is 10
Input 1 to use the default value, 2 to change it ([1:2], <cr>=1):
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): drive.tex
Input 1 for a vertical tree, 2 for a sideways tree ([1:2], <cr>=1):
Input 1 to include node numbers, 2 to omit them ([1:2], <cr>=1):
Input 1 to number all nodes, 2 to number leaves only ([1:2], <cr>=1):
Input 1 to color terminal nodes, 2 otherwise ([1:2], <cr>=1):
Choose amount of detail in nodes of LaTeX tree diagram
Input 0 for #errors, 1 for class sizes, 2 for nothing ([0:2], <cr>=1):
You can store the variables and/or values used to split and fit in a file
Choose 1 to skip this step, 2 to store split and fit variables,
3 to store split variables and their values
Input your choice ([1:3], <cr>=1):
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
```

```
Input 2 to save terminal node IDs for importance scoring; 1 otherwise ([1:2], <cr>=1):
Input file is created!
```

Results

```
Classification tree
Pruning by cross-validation
Data description file: drivedsc.txt
Training sample file: drive.txt
Missing value code: *
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is Drive
Number of records in data file: 428
Length of longest data entry: 26
Number of classes =  3
Class      #Cases    Proportion
4wd            94    0.21962617
fwd           224    0.52336449
rwd           110    0.25700935

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
 Column  Name            Minimum      Maximum   #Categories       #Missing
      3  Make       c                                    38
      5  Type       c                                     6
      6  Drive      d                                     3
     14  Rprice     s   1.0280E+04   1.9246E+05
     15  Dcost      s   9.8750E+03   1.7356E+05
     16  Enginsz    s   1.3000E+00   8.3000E+00
     17  Cylin      s  -1.0000E+00   1.2000E+01
     18  Hp         s   7.3000E+01   5.0000E+02
     19  City       s   1.0000E+01   6.0000E+01
     20  Hwy        s   1.2000E+01   6.6000E+01
     21  Weight     s   1.8500E+03   7.1900E+03
     22  Whlbase    s   8.9000E+01   1.4400E+02
     23  Length     s   1.4300E+02   2.2800E+02
     24  Width      s   6.4000E+01   8.1000E+01

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals    #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       428          0          0        10        0        0       11        0        2
No. cases used for training: 428
```

```
Univariate split highest priority
Interaction splits 2nd priority; no linear splits
Pruning by v-fold cross-validation, with v =  10
Selected tree is based on mean of CV estimates
Nearest-neighbor node models
Univariate preference
Estimated priors
Unit misclassification costs
Split values for N and S variables based on exhaustive search
Max number of split levels =  10
Minimum node size =  10

Pruning sequence
  Subtree     Pruned  #Terminal        True           Geometric
   number       node      nodes        alpha             mean
        0          0         20       0.0000          0.0000
        1          8         19       0.0000          0.0000
        2         44         18       0.0000          0.0000
        3         46         17       0.0000          0.0000
        4         12         16       0.0000          0.0000
        5         27         15       0.0000          0.0000
        6         13         14       0.0000          0.0000
        7         14         13       0.0000          0.0000
        8         30         12       0.0000          0.0000
        9         15         11       0.0000          0.0000
       10         23         10      0.86736E-18     0.17347E-17
       11          4          9      0.34694E-17     0.10396E-09
       12          5          6      0.31153E-02     0.76308E-02
       13          7          5      0.18692E-01     0.21918E-01
       14          2          4      0.25701E-01     0.25701E-01
       15          3          2      0.25701E-01     0.30996E-01
       16          1          1      0.37383E-01     0.17977+309

Number of SE's for pruned tree =   5.0000E-01

Size and CV mean cost and SE of subtrees:
 Tree   #Tnodes  Mean Cost   SE(Mean)   BSE(Mean)  Median Cost  BSE(Median)
   1       19   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   2       18   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   3       17   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   4       16   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   5       15   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   6       14   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   7       13   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   8       12   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   9       11   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
```

```
   10       10   2.500E-01   2.093E-02   1.911E-02   2.472E-01   3.174E-02
   11        9   2.500E-01   2.093E-02   1.911E-02   2.472E-01   3.174E-02
   12        6   2.360E-01   2.052E-02   2.306E-02   2.356E-01   4.397E-02
   13**      5   2.150E-01   1.986E-02   1.489E-02   2.093E-01   1.688E-02
   14        4   2.360E-01   2.052E-02   1.908E-02   2.381E-01   2.422E-02
   15        2   2.827E-01   2.177E-02   1.288E-02   2.821E-01   2.194E-02
   16        1   2.944E-01   2.203E-02   1.716E-02   2.907E-01   2.287E-02

 0-SE tree based on mean is marked with *
 0-SE tree based on median is marked with +
 Selected-SE tree based on mean using naive SE is marked with **
 Selected-SE tree based on mean using bootstrap SE is marked with --
 Selected-SE tree based on median and bootstrap SE is marked with ++
 ** tree and ++ tree are the same


 Following tree is based on mean CV with naive SE estimate (**).

 Structure of final tree. Each terminal node is marked with a T.

 Node cost is node misclassification cost divided by number of training cases
        Node    Total    Train    Predicted        Node    Split variable followed by
       label    cases    cases    class            cost    (+)fit variable(s)
           1      428      428    fwd          2.874E-01 Make +Make
           2      151      151    rwd          2.914E-01 Make +Make
          4T       39       39    4wd          2.308E-01 Hwy +Hwy
          5T      112      112    rwd          2.143E-01 Type +Type
           3      277      277    fwd          2.274E-01 Type +Type
           6       73       73    4wd          4.384E-01 Type +Type
         12T       36       36    rwd          3.889E-01 Type +Type
         13T       37       37    4wd          1.622E-01 Make +Make
          7T      204      204    fwd          1.029E-01 Hwy +Hwy
  The variables preceded with a + sign are those used in the nearest neighbor models.
 Number of terminal nodes of final tree: 5
 Total number of nodes of final tree:    9

 Classification tree:

  Node 1: Make = "Audi", "BMW", "GMC", "Hummer", "Infiniti", "Isuzu",
                 "Jaguar", "Jeep", "Land-Rover", "Lexus", "Lincoln", "Mazda",
                 "Mercedes", "Porsche", "Subaru"
    Node 2: Make = "Audi", "Hummer", "Isuzu", "Jeep", "Land-Rover", "Subaru"
      Node 4: Mean cost =  2.30769E-01
    Node 2: Make /= "Audi", "Hummer", "Isuzu", "Jeep", "Land-Rover", "Subaru"
      Node 5: Mean cost =  2.14286E-01
  Node 1: Make /= "Audi", "BMW", "GMC", "Hummer", "Infiniti", "Isuzu",
```

```
                "Jaguar", "Jeep", "Land-Rover", "Lexus", "Lincoln", "Mazda",
                "Mercedes", "Porsche", "Subaru"
   Node 3: Type = "pickup", "sports", "suv"
     Node 6: Type = "pickup", "sports"
       Node 12: Mean cost =  3.88889E-01
     Node 6: Type /= "pickup", "sports"
       Node 13: Mean cost =  1.62162E-01
   Node 3: Type /= "pickup", "sports", "suv"
     Node 7: Mean cost =  1.02941E-01

***************************************************************


Node 1: Intermediate node
A case goes into Node 2 if Make =
 "Audi", "BMW", "GMC", "Hummer", "Infiniti", "Isuzu", "Jaguar", "Jeep",
 "Land-Rover", "Lexus", "Lincoln", "Mazda", "Mercedes", "Porsche", "Subaru"
Nearest-neighbor K =           7
Make mode = Toyota
                               Fit variable
Class      Number  ClassPrior  Make
4wd            94     0.21963
fwd           224     0.52336
rwd           110     0.25701
Number of training cases misclassified =  123
If node model is inapplicable due to missing values, predicted class  =
fwd
```

Although the number of nearest neighbors is 7 in this node, the neighborhood is the entire node because the fit variable, Type, is categorical.

```
----------------------------
Node 2: Intermediate node
A case goes into Node 4 if Make =
 "Audi", "Hummer", "Isuzu", "Jeep", "Land-Rover", "Subaru"
Nearest-neighbor K =           6
Make mode = Mercedes
                               Fit variable
Class      Number  ClassPrior  Make
4wd            56     0.37086
fwd            18     0.11921
rwd            77     0.50993
Number of training cases misclassified =  44
If node model is inapplicable due to missing values, predicted class  =
rwd
----------------------------
Node 4: Terminal node
Nearest-neighbor K =  4
```

```
Hwy mean =  2.4564E+01
                               Fit variable
Class      Number  ClassPrior  Hwy
4wd            30     0.76923
fwd             9     0.23077
rwd             0     0.00000
----------------------------
Node 5: Terminal node
Nearest-neighbor K =  5
Type mode = car
                               Fit variable
Class      Number  ClassPrior  Type
4wd            26     0.23214
fwd             9     0.08036
rwd            77     0.68750
----------------------------
Node 3: Intermediate node
A case goes into Node 6 if Type =
 "pickup", "sports", "suv"
Nearest-neighbor K =            6
Type mode = car
                               Fit variable
Class      Number  ClassPrior  Type
4wd            38     0.13718
fwd           206     0.74368
rwd            33     0.11913
Number of training cases misclassified =  63
If node model is inapplicable due to missing values, predicted class  =
fwd
----------------------------
Node 6: Intermediate node
A case goes into Node 12 if Type =
 "pickup", "sports"
Nearest-neighbor K =            5
Type mode = suv
                               Fit variable
Class      Number  ClassPrior  Type
4wd            28     0.38356
fwd            23     0.31507
rwd            22     0.30137
Number of training cases misclassified =  32
If node model is inapplicable due to missing values, predicted class  =
4wd
----------------------------
Node 12: Terminal node
Nearest-neighbor K =  4
```

```
Type mode = sports
                                Fit variable
Class      Number  ClassPrior  Type
4wd             8     0.22222
fwd             6     0.16667
rwd            22     0.61111
----------------------------
Node 13: Terminal node
Nearest-neighbor K =  4
Make mode = Toyota
                                Fit variable
Class      Number  ClassPrior  Make
4wd            20     0.54054
fwd            17     0.45946
rwd             0     0.00000
----------------------------
Node 7: Terminal node
Nearest-neighbor K =  6
Hwy mean =  3.0191E+01
                                Fit variable
Class      Number  ClassPrior  Hwy
4wd            10     0.04902
fwd           183     0.89706
rwd            11     0.05392
----------------------------


Classification matrix for training sample:
Predicted      True class
class            4wd       fwd       rwd
4wd               60        17         0
fwd               11       195        11
rwd               23        12        99
Total             94       224       110

Number of cases used for tree construction =  428
Number misclassified =  74
Resubstitution est. of mean misclassification cost =   0.17289719626168223

LaTeX code for tree is in drive.tex
```

Figure 3 shows the tree model, which misclassifies 74 observations.

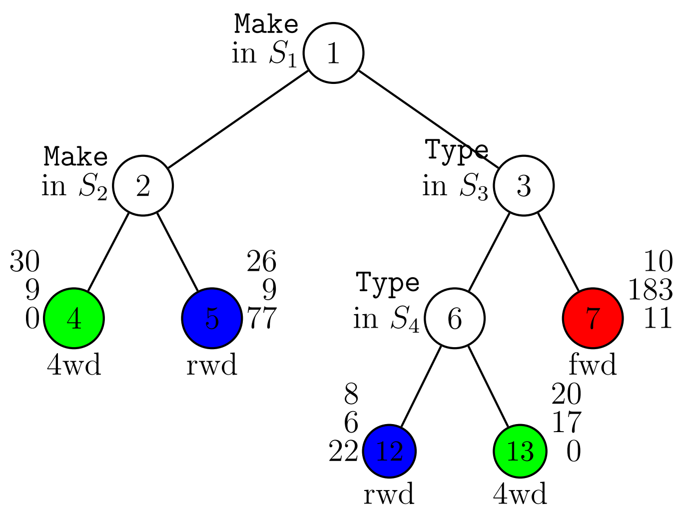

Figure 3: GUIDE 0.50-SE classification tree for predicting `Drive` using univariate nearest-neighbor node models, estimated priors and unit misclassification costs. At each split, an observation goes to the left branch if and only if the condition is satisfied. Set S_1 = {Audi, BMW, GMC, Hummer, Infiniti, Isuzu, Jaguar, Jeep, Land-Rover, Lexus, Lincoln, Mazda, Mercedes, Porsche, Subaru}. Set S_2 = {Audi, Hummer, Isuzu, Jeep, Land-Rover, Subaru}. Set S_3 = {pickup, sports, suv}. Set S_4 = {pickup, sports}. Predicted classes (based on estimated misclassification cost) printed below terminal nodes; sample sizes for `Drive` = 4wd, fwd, and rwd, respectively, beside nodes.

4.2.5 Kernel density estimates: car data

An alternative to nearest-neighbor models is kernel discrimination models, where classification is based on maximum likelihood with class densities estimated by the kernel method. Unlike nearest-neighbor, however, this option also yields an estimated class probability vector for each observation. Therefore it can serve as a nonparametric alternative to multinomial logistic regression. Empirical evidence indicates that the nearest-neighbor and kernel methods possess similar prediction accuracy. See Loh (2009) for more details. Following is a log of the input file generation step for the kernel method.

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice: 1
Name of batch input file: driveker.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: driveker.out
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1): 2
Input 1 for simple, 2 for nearest-neighbor, 3 for kernel method ([1:3], <cr>=1): 3
 This is where you choose kernel density estimation.
Input 1 for univariate, 2 for bivariate preference ([1:2], <cr>=1):
Input 1 for interaction tests, 2 to skip them ([1:2], <cr>=1):
Input 1 to prune by CV, 2 by test sample, 3 for no pruning ([1:3], <cr>=1):

Input name of data description file (maximum 100 characters; enclose within quotes
if it contains spaces or non alphanumeric characters): drivedsc.txt
Reading data description file ...
Training sample file: drive.txt
Missing value code: *
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is Drive
Length of longest data entry = 26
Total number of cases =         428
Number of classes =            3
Col. no. Categorical variable    #levels    #missing values
       3 Make                         38                  0
       5 Type                          6                  0
Checking data ...
Class name          Num. cases   Proportion
4wd                         94   0.21962617
fwd                        224   0.52336449
```

```
rwd                         110   0.25700935
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       428          0          0       10        0        0       11        0        2
No. cases used for training =          428
Default number of cross-validations =          10
Input 1 to accept the default, 2 to change it ([1:2], <cr>=1):
Best tree may be chosen based on mean or median CV estimate
Input 1 for mean-based, 2 for median-based ([1:2], <cr>=1):
Input number of SEs for pruning ([0.00:1000.00], <cr>=0.50):
Choose 1 for estimated priors, 2 for equal priors, 3 to input the priors from a file
Input 1, 2, or 3 ([1:3], <cr>=1):
Choose 1 for unit misclassification costs, 2 to input costs from a file
Input 1 or 2 ([1:2], <cr>=1):
Choose a split point selection method for numerical variables:
Choose 1 to use faster method based on sample quantiles
Choose 2 to use exhaustive search
Input 1 or 2 ([1:2], <cr>=2):
Default max number of split levels =          10
Input 1 to accept this value, 2 to change it ([1:2], <cr>=1):
Default minimum node sample size is 10
Input 1 to use the default value, 2 to change it ([1:2], <cr>=1):
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): driveker.tex
Input 1 for a vertical tree, 2 for a sideways tree ([1:2], <cr>=1):
Input 1 to include node numbers, 2 to omit them ([1:2], <cr>=1):
Input 1 to number all nodes, 2 to number leaves only ([1:2], <cr>=1):
Input 1 to color terminal nodes, 2 otherwise ([1:2], <cr>=1):
Choose amount of detail in nodes of LaTeX tree diagram
Input 0 for #errors, 1 for class sizes, 2 for nothing ([0:2], <cr>=1):
You can store the variables and/or values used to split and fit in a file
Choose 1 to skip this step, 2 to store split and fit variables,
3 to store split variables and their values
Input your choice ([1:3], <cr>=1):
Input 2 to save fitted values and node IDs, 1 otherwise ([1:2], <cr>=1): 2
Input name of file to store node IDs and fitted values: driveker.fit
 This file contains the predicted class and terminal node label for each observation.
Input 2 to save terminal node IDs for importance scoring; 1 otherwise ([1:2], <cr>=1):
Input name of file to store predicted class and probability: driveker.pro
 This file contains the estimated class probabilities for each observation.
Input file is created!
Run GUIDE with the command: guide < driveker.in
```

The results in the output file are given next.

```
Classification tree
```

```
Pruning by cross-validation
Data description file: drivedsc.txt
Training sample file: drive.txt
Missing value code: *
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is Drive
Number of records in data file: 428
Length of longest data entry: 26
Number of classes =  3
Class      #Cases    Proportion
4wd            94    0.21962617
fwd           224    0.52336449
rwd           110    0.25700935

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
 Column  Name            Minimum      Maximum   #Categories       #Missing
      3  Make       c                                    38
      5  Type       c                                     6
      6  Drive      d                                     3
     14  Rprice     s   1.0280E+04   1.9246E+05
     15  Dcost      s   9.8750E+03   1.7356E+05
     16  Enginsz    s   1.3000E+00   8.3000E+00
     17  Cylin      s  -1.0000E+00   1.2000E+01
     18  Hp         s   7.3000E+01   5.0000E+02
     19  City       s   1.0000E+01   6.0000E+01
     20  Hwy        s   1.2000E+01   6.6000E+01
     21  Weight     s   1.8500E+03   7.1900E+03
     22  Whlbase    s   8.9000E+01   1.4400E+02
     23  Length     s   1.4300E+02   2.2800E+02
     24  Width      s   6.4000E+01   8.1000E+01

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       428          0          0       10        0        0       11        0        2
No. cases used for training: 428

Univariate split highest priority
Interaction splits 2nd priority; no linear splits
Pruning by v-fold cross-validation, with v =  10
Selected tree is based on mean of CV estimates
Kernel density node models
Univariate preference
Estimated priors
```

```
Unit misclassification costs
Split values for N and S variables based on exhaustive search
Max number of split levels =  10
Minimum node size =  10

Pruning sequence
  Subtree     Pruned  #Terminal        True           Geometric
   number       node      nodes        alpha              mean
        0          0         20       0.0000          0.0000
        1          8         19       0.0000          0.0000
        2         22         17       0.0000          0.0000
        3         46         16       0.0000          0.0000
        4         12         15       0.0000          0.0000
        5         27         14       0.0000          0.0000
        6         13         13       0.0000          0.0000
        7         14         12       0.0000          0.0000
        8         30         11       0.0000          0.0000
        9         15         10       0.0000          0.0000
       10          4          9      0.34694E-17     0.90034E-10
       11         23          8      0.23364E-02     0.23364E-02
       12          5          6      0.23364E-02     0.61817E-02
       13          7          5      0.16355E-01     0.20502E-01
       14          2          4      0.25701E-01     0.25701E-01
       15          3          2      0.25701E-01     0.30996E-01
       16          1          1      0.37383E-01     0.17977+309

Number of SE's for pruned tree =   5.0000E-01

Size and CV mean cost and SE of subtrees:
 Tree   #Tnodes  Mean Cost   SE(Mean)   BSE(Mean)  Median Cost  BSE(Median)
   1       19   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   2       17   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   3       16   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   4       15   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   5       14   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   6       13   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   7       12   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   8       11   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
   9       10   2.640E-01   2.131E-02   1.632E-02   2.705E-01   2.220E-02
  10        9   2.570E-01   2.112E-02   1.607E-02   2.558E-01   2.051E-02
  11        8   2.523E-01   2.100E-02   1.982E-02   2.558E-01   2.311E-02
  12        6   2.477E-01   2.086E-02   1.913E-02   2.470E-01   3.444E-02
  13**      5   2.266E-01   2.024E-02   1.432E-02   2.118E-01   2.304E-02
  14        4   2.687E-01   2.143E-02   1.343E-02   2.674E-01   2.303E-02
  15        2   2.827E-01   2.177E-02   1.288E-02   2.821E-01   2.194E-02
  16        1   2.944E-01   2.203E-02   1.716E-02   2.907E-01   2.287E-02
```

```
0-SE tree based on mean is marked with *
0-SE tree based on median is marked with +
Selected-SE tree based on mean using naive SE is marked with **
Selected-SE tree based on mean using bootstrap SE is marked with --
Selected-SE tree based on median and bootstrap SE is marked with ++
** tree and ++ tree are the same


Following tree is based on mean CV with naive SE estimate (**).

Structure of final tree. Each terminal node is marked with a T.

Node cost is node misclassification cost divided by number of training cases
       Node    Total    Train    Predicted         Node    Split variable followed by
      label    cases    cases    class             cost    (+)fit variable(s)
          1      428      428    fwd           2.874E-01 Make +Make
          2      151      151    rwd           2.914E-01 Make +Make
         4T       39       39    4wd           2.308E-01 Hwy +Hwy
         5T      112      112    rwd           2.143E-01 Type +Type
          3      277      277    fwd           2.274E-01 Type +Type
          6       73       73    4wd           4.384E-01 Type +Type
        12T       36       36    rwd           3.889E-01 Type +Type
        13T       37       37    4wd           1.622E-01 Make +Make
         7T      204      204    fwd           1.029E-01 Hwy +Hwy
```

In the above, ''split variable'' refers to the variable selected to split the node and ''fit variable(s)'' refers to the one(s) used to estimate the class kernel densities. Fit variables are indicated with a preceding + sign. In this example, the split and fit variables are the same in every node.
If a categorical variable (e.g., Type) is selected for fitting, discrete kernel density estimates are used. A dash (-) indicates that a node is not split, usually due to sample size being too small, in which case all the observations in the node are predicted as belonging to the class that minimizes the misclassification cost.

```
Number of terminal nodes of final tree: 5
Total number of nodes of final tree:    9

Classification tree:

 Node 1: Make = "Audi", "BMW", "GMC", "Hummer", "Infiniti", "Isuzu",
               "Jaguar", "Jeep", "Land-Rover", "Lexus", "Lincoln", "Mazda",
               "Mercedes", "Porsche", "Subaru"
   Node 2: Make = "Audi", "Hummer", "Isuzu", "Jeep", "Land-Rover", "Subaru"
     Node 4: Mean cost =  2.30769E-01
   Node 2: Make /= "Audi", "Hummer", "Isuzu", "Jeep", "Land-Rover", "Subaru"
     Node 5: Mean cost =  2.14286E-01
```

```
 Node 1: Make /= "Audi", "BMW", "GMC", "Hummer", "Infiniti", "Isuzu",
                "Jaguar", "Jeep", "Land-Rover", "Lexus", "Lincoln", "Mazda",
                "Mercedes", "Porsche", "Subaru"
   Node 3: Type = "pickup", "sports", "suv"
     Node 6: Type = "pickup", "sports"
       Node 12: Mean cost =  3.88889E-01
     Node 6: Type /= "pickup", "sports"
       Node 13: Mean cost =  1.62162E-01
   Node 3: Type /= "pickup", "sports", "suv"
     Node 7: Mean cost =  1.02941E-01

***************************************************************


Node 1: Intermediate node
A case goes into Node 2 if Make =
 "Audi", "BMW", "GMC", "Hummer", "Infiniti", "Isuzu", "Jaguar", "Jeep",
 "Land-Rover", "Lexus", "Lincoln", "Mazda", "Mercedes", "Porsche", "Subaru"
Make mode = Toyota
                                Bandwidth
Class      Number  ClassPrior   Make
4wd            94     0.21963
fwd           224     0.52336
rwd           110     0.25701
Number of training cases misclassified =  123
If node model is inapplicable due to missing values, predicted class  =
fwd
```

Categorical variables, such as Make, do not have bandwidths. Their kernel density estimates are the sample cell frequencies.

```
----------------------------
Node 2: Intermediate node
A case goes into Node 4 if Make =
 "Audi", "Hummer", "Isuzu", "Jeep", "Land-Rover", "Subaru"
Make mode = Mercedes
                                Bandwidth
Class      Number  ClassPrior   Make
4wd            56     0.37086
fwd            18     0.11921
rwd            77     0.50993
Number of training cases misclassified =  44
If node model is inapplicable due to missing values, predicted class  =
rwd
----------------------------
Node 4: Terminal node
Hwy mean =  2.4564E+01
                                Bandwidth
```

```
Class      Number  ClassPrior   Hwy
4wd            30     0.76923   5.3552E+00
fwd             9     0.23077   5.3739E+00
rwd             0     0.00000   0.0000E+00
 The numbers in the last column give the kernel density bandwidth for
 each class.
----------------------------
Node 5: Terminal node
Type mode = car
                                Fit variable
Class      Number  ClassPrior   Type
4wd            26     0.23214
fwd             9     0.08036
rwd            77     0.68750
----------------------------
Node 3: Intermediate node
A case goes into Node 6 if Type =
 "pickup", "sports", "suv"
Type mode = car
                                Bandwidth
Class      Number  ClassPrior   Type
4wd            38     0.13718
fwd           206     0.74368
rwd            33     0.11913
Number of training cases misclassified =  63
If node model is inapplicable due to missing values, predicted class  =
fwd
----------------------------
Node 6: Intermediate node
A case goes into Node 12 if Type =
 "pickup", "sports"
Type mode = suv
                                Bandwidth
Class      Number  ClassPrior   Type
4wd            28     0.38356
fwd            23     0.31507
rwd            22     0.30137
Number of training cases misclassified =  32
If node model is inapplicable due to missing values, predicted class  =
4wd
----------------------------
Node 12: Terminal node
Type mode = sports
                                Fit variable
Class      Number  ClassPrior   Type
4wd             8     0.22222
```

```
fwd             6     0.16667
rwd            22     0.61111
----------------------------
Node 13: Terminal node
Make mode = Toyota
                                Fit variable
Class      Number  ClassPrior   Make
4wd            20     0.54054
fwd            17     0.45946
rwd             0     0.00000
----------------------------
Node 7: Terminal node
Hwy mean =  3.0191E+01
                                Bandwidth
Class      Number  ClassPrior   Hwy
4wd            10     0.04902   5.8465E+00
fwd           183     0.89706   3.9227E+00
rwd            11     0.05392   5.4617E+00
----------------------------


Classification matrix for training sample:
Predicted      True class
class            4wd       fwd       rwd
4wd               60        17         0
fwd               11       195        11
rwd               23        12        99
Total             94       224       110

Number of cases used for tree construction =  428
Number misclassified =  74
Resubstitution est. of mean misclassification cost =   0.17289719626168223

Predicted class probability estimates are stored in driveker.pro
Observed and fitted values are stored in driveker.fit
LaTeX code for tree is in driveker.tex
```

The tree is the same as that in Figure 3. The top several lines of the file `driveker.fit`, which contains the terminal node label and predicted class for each observation in the training sample, are:

```
train         node  observed  predicted
    y            7    "fwd"  "fwd"
    y            7    "fwd"  "fwd"
    y            7    "fwd"  "fwd"
    y            7    "fwd"  "fwd"
```

```
y            7    "fwd"  "fwd"
y            7    "fwd"  "fwd"
y            7    "fwd"  "fwd"
y            7    "fwd"  "fwd"
```

The corresponding lines of the file `driveker.pro`, giving the estimated class probabilities for each observation, are:

```
 "4wd" "fwd" "rwd"    predicted  observed
 0.02241 0.95299 0.02460  "fwd"  "fwd"
 0.02241 0.95299 0.02460  "fwd"  "fwd"
 0.01701 0.96092 0.02207  "fwd"  "fwd"
 0.01701 0.96092 0.02207  "fwd"  "fwd"
 0.01701 0.96092 0.02207  "fwd"  "fwd"
 0.01874 0.95887 0.02239  "fwd"  "fwd"
 0.01874 0.95887 0.02239  "fwd"  "fwd"
 0.02437 0.94910 0.02653  "fwd"  "fwd"
```

5 Regression

5.1 Stepwise least-squares

We use the baseball dataset `bbdat.txt` to show the results for regression trees when there are no missing values. The data give the log-salary and performance measures of 263 professional baseball players (Hoaglin and Velleman, 1995). The response variable is the logarithm of salary (`Logsalary`). The data description file `bbdsc.txt` consists of the following lines:

```
bbdat.txt
NA
1
1 Id x
2 Name x
3 Bat86 n
4 Hit86 n
5 Hr86 n
6 Run86 n
7 Rb86 n
8 Wlk86 n
9 Yrs n
10 Batcr n
11 Hitcr n
12 Hrcr n
```

```
13 Runcr n
14 Rbcr n
15 Wlkcr n
16 League86 b
17 Div86 b
18 Team86 c
19 Pos86 b
20 Puto86 n
21 Asst86 n
22 Err86 n
23 Salary x
24 League87 b
25 Team87 c
26 Logsalary d
```

Notice that there are four variables having the "b" variable type. This means that 0-1 dummy variables will be created for them in fitting the node linear models. The following shows how the input file is created.

```
 0. Read the warranty disclaimer
 1. Create an input file for batch run
 2. Fit a model without creating input file
 3. Convert data to other formats
 4. Variable importance scoring and differential item functioning
 Input your choice: 1
 Name of batch input file: stepin.txt
 Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
 Name of batch output file: stepout.txt
 Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
 Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
 Choose type of regression model:
  1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
  5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
 Input choice ([1:6], <cr>=1):
 Input 1 for least squares, 2 least median of squares ([1:2], <cr>=1):
 Choose complexity of model to use at each node:
 0: stepwise linear, 1: multiple linear, 2: best polynomial, 3: constant,
 4: stepwise simple ANCOVA ([0:4], <cr>=0):
 Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

 Input name of data description file (maximum 100 characters; enclose within quotes
 if it contains spaces or non alphanumeric characters): bbdsc.txt
 Reading data description file ...
 Training sample file: bbdat.txt
 Missing value code: NA
 Records in data file start on line 1
```

```
Dependent variable is Logsalary
Length of longest data entry = 17
Total number of cases =        263
Col. no. Categorical variable    #levels    #missing values
      16 Leag86                        2                  0
      17 Div86                         2                  0
      18 Team86                       24                  0
      19 Pos86                        23                  0
      24 Leag87                        2                  0
      25 Team87                       24                  0
Checking data ...
The program will try to create the variables in the description file.
If it is unsuccessful, please create the columns yourself...
Number of dummy variables created:             25
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       263          0          0        3       16        0        0        4        2
No weight variable in data file
No. cases used for training =          263
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): step.tex
Input 2 to save fitted values and node IDs, 1 otherwise ([1:2], <cr>=1): 2
Input name of file to store node IDs and fitted values: step.fit
Input file is created!
```

The contents from the file `stepout.txt` follow. They show a tree with two terminal nodes and give the regression coefficients, sample means of the dependent and predictor variables, MSE and R^2 values, and names of the split variables in each node.

```
Least squares regression tree
Predictions truncated at global min and max of D sample values
 The predicted values are truncated at the minimum and
    maximum values of the training sample by default.
Pruning by cross-validation
Data description file: bbdsc.txt
Training sample file: bbdat.txt
Missing value code: NA
Records in data file start on line 1
Dependent variable is Logsalary
Piecewise forward and backward stepwise regression
F-to-enter and F-to-delete =    4.0000000000000000  3.9900000000000002
 These default F values are the same as those used in SAS.
Using as many variables as needed
Number of records in data file: 263
```

```
Length of longest data entry = 17
Number of dummy variables created =  25

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
 Column  Name            Minimum      Maximum   #Categories       #Missing
      3  Bat86      n   1.2700E+02   6.8700E+02
      4  Hit86      n   3.2000E+01   2.3800E+02
      5  Hr86       n   0.0000E+00   4.0000E+01
      6  Run86      n   1.3000E+01   1.3000E+02
      7  Rb86       n   8.0000E+00   1.2100E+02
      8  Wlk86      n   3.0000E+00   1.0500E+02
      9  Yrs        n   1.0000E+00   2.4000E+01
     10  Batcr      n   1.8100E+02   1.4053E+04
     11  Hitcr      n   4.2000E+01   4.2560E+03
     12  Hrcr       n   0.0000E+00   5.4800E+02
     13  Runcr      n   1.8000E+01   2.1650E+03
     14  Rbcr       n   9.0000E+00   1.6590E+03
     15  Wlkcr      n   8.0000E+00   1.5660E+03
     16  Leag86     b                                      2
     17  Div86      b                                      2
     18  Team86     c                                     24
     19  Pos86      b                                     23
     20  Puto86     n   0.0000E+00   1.3770E+03
     21  Asst86     n   0.0000E+00   4.9200E+02
     22  Err86      n   0.0000E+00   3.2000E+01
     24  Leag87     b                                      2
     25  Team87     c                                     24
     26  Logsalary  d   4.2121E+00   7.8079E+00
  =================== Constructed variables ===================
   The F variables below this line are dummy variables constructed from the B variables.
     27  Leag8.N    f   0.0000E+00   1.0000E+00
     28  Div86.W    f   0.0000E+00   1.0000E+00
     29  Pos86.1O   f   0.0000E+00   1.0000E+00
     30  Pos86.23   f   0.0000E+00   1.0000E+00
     31  Pos86.2B   f   0.0000E+00   1.0000E+00
     32  Pos86.2S   f   0.0000E+00   1.0000E+00
     33  Pos86.32   f   0.0000E+00   1.0000E+00
     34  Pos86.3B   f   0.0000E+00   1.0000E+00
     35  Pos86.3O   f   0.0000E+00   1.0000E+00
     36  Pos86.3S   f   0.0000E+00   1.0000E+00
     37  Pos86.C    f   0.0000E+00   1.0000E+00
     38  Pos86.CD   f   0.0000E+00   1.0000E+00
     39  Pos86.CF   f   0.0000E+00   1.0000E+00
     40  Pos86.DH   f   0.0000E+00   1.0000E+00
```

```
     41  Pos86.DO   f   0.0000E+00   1.0000E+00
     42  Pos86.LF   f   0.0000E+00   1.0000E+00
     43  Pos86.O1   f   0.0000E+00   1.0000E+00
     44  Pos86.OD   f   0.0000E+00   1.0000E+00
     45  Pos86.OF   f   0.0000E+00   1.0000E+00
     46  Pos86.OS   f   0.0000E+00   1.0000E+00
     47  Pos86.RF   f   0.0000E+00   1.0000E+00
     48  Pos86.S3   f   0.0000E+00   1.0000E+00
     49  Pos86.SS   f   0.0000E+00   1.0000E+00
     50  Pos86.UT   f   0.0000E+00   1.0000E+00
     51  Leag8.N    f   0.0000E+00   1.0000E+00

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       263          0          0        3       16        0        0        4        2
No weight variable in data file
No. cases used for training: 263

Missing values imputed with node means for model fitting
```

The default method of handling missing values is node-mean imputation.

```
Interaction tests on all variables
Pruning by v-fold cross-validation, with v =  10
Selected tree is based on mean of CV estimates
Fraction of cases used for splitting each node =   1.0000
Max number of split levels =  10
Minimum node size =  13

Pruning sequence
  Subtree     Pruned  #Terminal        True           Geometric
   number       node      nodes        alpha             mean
        0          0         15       0.0000          0.0000
        1          8         14       0.0000          0.0000
        2         19         13       0.0000          0.0000
        3         30         12       0.0000          0.0000
        4         37         11     0.85255E-03     0.85343E-03
        5         18         10     0.85431E-03     0.13218E-02
        6          4          8     0.20451E-02     0.24625E-02
        7          5          7     0.29651E-02     0.34926E-02
        8          7          4     0.41140E-02     0.46348E-02
        9          2          3     0.52216E-02     0.75400E-02
       10          3          2     0.10888E-01     0.44658E-01
       11          1          1     0.18317         0.17977+309

Number of SE's for pruned tree =   5.0000E-01

Size and CV MSE and SE of subtrees:
```

```
 Tree   #Tnodes  Mean MSE   SE(Mean)    BSE(Mean)  Median MSE  BSE(Median)
   1       14   2.039E-01   2.220E-02   2.234E-02   1.971E-01   2.132E-02
   2       13   2.039E-01   2.220E-02   2.234E-02   1.971E-01   2.132E-02
   3       12   2.039E-01   2.220E-02   2.234E-02   1.971E-01   2.132E-02
   4       11   1.814E-01   2.085E-02   2.026E-02   1.704E-01   3.624E-02
   5       10   1.790E-01   2.099E-02   1.972E-02   1.601E-01   3.561E-02
   6        8   1.858E-01   2.103E-02   1.996E-02   1.707E-01   3.453E-02
   7        7   1.896E-01   2.134E-02   1.986E-02   1.737E-01   3.174E-02
   8        4   1.780E-01   2.115E-02   2.206E-02   1.457E-01   3.856E-02
   9        3   1.585E-01   1.949E-02   1.546E-02   1.439E-01   2.647E-02
  10**      2   1.208E-01   1.439E-02   1.378E-02   1.166E-01   1.884E-02
  11        1   3.469E-01   2.575E-02   2.224E-02   3.456E-01   3.877E-02

0-SE tree based on mean is marked with *
0-SE tree based on median is marked with +
Selected-SE tree based on mean using naive SE is marked with **
Selected-SE tree based on mean using bootstrap SE is marked with --
Selected-SE tree based on median and bootstrap SE is marked with ++
** tree and ++ tree are the same


Following tree is based on mean CV with naive SE estimate (**).

Structure of final tree. Each terminal node is marked with a T.

D-mean is mean of Logsalary in the node
Cases fit give the number of cases used to fit node
MSE and R^2 are based on all cases in node
       Node    Total    Cases Matrix    Node      Node      Node  Split          Other
       label   cases      fit  rank    D-mean     MSE       R^2   variable       variables
          1      263      263     9  5.945E+00  2.907E-01  0.6391 Yrs
          2T     143      143     7  5.506E+00  8.336E-02  0.8907 Yrs
          3T     120      120     6  6.469E+00  1.258E-01  0.6456 Bat86

Number of terminal nodes of final tree: 2
Total number of nodes of final tree:    3

Regression tree:

 Node 1: Yrs <=    6.00000 or NA
   Node 2: Logsalary-mean =   5.50632
 Node 1: Yrs >    6.00000 and not NA
   Node 3: Logsalary-mean =   6.46866

***************************************************************
```

```
Node 1: Intermediate node
A case goes into Node 2 if Yrs <=  6.0000000E+00 or NA
Yrs mean =  7.3802E+00
Coefficients of least squares regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   4.1321E+00       28.07 0.0000
Bat86     -2.4528E-03       -2.73 0.0068  1.2700E+02  4.0829E+02  6.8700E+02
Hit86      1.4558E-02        5.16 0.0000  3.2000E+01  1.0916E+02  2.3800E+02
Wlk86      1.0020E-02        3.82 0.0002  3.0000E+00  4.1722E+01  1.0500E+02
Yrs        7.0500E-02        4.24 0.0000  1.0000E+00  7.3802E+00  2.4000E+01
Runcr      1.1939E-03        3.11 0.0021  1.8000E+01  3.6808E+02  2.1650E+03
Wlkcr     -9.6467E-04       -2.20 0.0291  8.0000E+00  2.6655E+02  1.5660E+03
Leag8.N    1.4081E-01        2.09 0.0373  0.0000E+00  4.7148E-01  1.0000E+00
Pos86.C    3.3729E-01        3.05 0.0025  0.0000E+00  1.1407E-01  1.0000E+00
Mean of Logsalary =   5.9454102795235446
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
----------------------------
Node 2: Terminal node
Coefficients of least squares regression functions:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   4.1385E+00       39.36 0.0000
Bat86     -1.8387E-03       -4.25 0.0000  1.5100E+02  4.1345E+02  6.8700E+02
Run86      1.5359E-02        6.46 0.0000  1.3000E+01  5.6699E+01  1.1900E+02
Yrs        1.1659E-01        4.86 0.0000  1.0000E+00  3.8042E+00  6.0000E+00
Batcr      5.0610E-04        5.44 0.0000  1.8100E+02  1.2046E+03  3.3740E+03
Rbcr       1.5954E-03        2.86 0.0049  9.0000E+00  1.4315E+02  4.7500E+02
Pos86.CF  -2.3212E-01       -2.74 0.0070  0.0000E+00  1.0490E-01  1.0000E+00
Mean of Logsalary =   5.5063156054013094
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
 The truncation limits are the minimum and maximum values of the entire training sample.
----------------------------
Node 3: Terminal node
Coefficients of least squares regression functions:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   6.2274E+00       33.81 0.0000
Hit86      4.7082E-03        4.73 0.0000  3.2000E+01  1.0759E+02  2.0000E+02
Yrs       -1.0408E-01       -6.16 0.0000  7.0000E+00  1.1642E+01  2.4000E+01
Runcr      7.9765E-04        3.26 0.0015  6.7000E+01  6.1369E+02  2.1650E+03
Rbcr       5.9958E-04        2.56 0.0118  8.2000E+01  5.6998E+02  1.6590E+03
Puto86     4.1481E-04        3.43 0.0008  0.0000E+00  2.7723E+02  1.3140E+03
Mean of Logsalary =   6.4686647661858760
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
----------------------------

Proportion of variance (R-squared) explained by tree model =  .8745
```

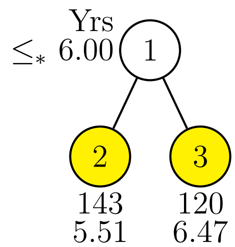

Figure 4: GUIDE piecewise linear least-squares regression tree with stepwise variable selection for predicting Logsalary. At each intermediate node, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Sample sizes and means of Logsalary printed below nodes.

```
Observed and fitted values are stored in step.fit
LaTeX code for tree is in step.tex
```

The LaTeX tree produced by the file `step.tex` is shown in Figure 4.

5.2 Least-squares simple polynomial

Often it is useful to be able to visualize the fitted regression function and the data simultaneously. This can be accomplished by fitting a piecewise simple linear model, where the best single regressor is selected to fit a straight line in each node, as follows.

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice: 1
Name of batch input file: linin.txt
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: linout.txt
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
Choose type of regression model:
 1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
 5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
Input choice ([1:6], <cr>=1):
Input 1 for least squares, 2 least median of squares ([1:2], <cr>=1):
Choose complexity of model to use at each node:
```

```
0: stepwise linear, 1: multiple linear, 2: best polynomial, 3: constant,
4: stepwise simple ANCOVA ([0:4], <cr>=0): 2
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):
 The default degree of the polynomial is 1.

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: bbdsc.txt
Reading data description file ...
Training sample file: bbdat.txt
Missing value code: NA
Records in data file start on line 1
Warning: B variables changed to C
 This warning is triggered because the description file contains some variables
 with the B designation, which is not allowed in piecewise polynomial regression.
Dependent variable is Logsalary
Reading data file ...
Number of records in data file: 263
Length of longest data entry: 17
Checking for missing values ...
Total number of cases: 263
Col. no. Categorical variable    #levels    #missing values
      16 Leag86                        2                  0
      17 Div86                         2                  0
      18 Team86                       24                  0
      19 Pos86                        23                  0
      24 Leag87                        2                  0
      25 Team87                       24                  0
Re-checking data ...
Assigning codes to categorical and missing values
firstln =           1
Finish checking data
Rereading data
     Total  #cases w/   #missing
    #cases     miss. D  ord. vals    #X-var    #N-var    #F-var    #S-var    #B-var    #C-var
       263           0          0         3        16         0         0         0         6
No weight variable in data file
No. cases used for training: 263
Finish reading data file
Rereading data
     Total  #cases w/   #missing
    #cases     miss. D  ord. vals    #X-var    #N-var    #F-var    #S-var    #B-var    #C-var
       263           0          0         3        16         0         0         0         6
No weight variable in data file
No. cases used for training: 263
Finish reading data file
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
```

```
Input file name to store LaTeX code (use .tex as suffix): lin.tex
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
Input name of file to store node ID and fitted value of each case: lin.fit
Input file is created!
Run GUIDE with the command: guide < linin.txt
```

Partial contents of `linout.txt`

```
Size and CV MSE and SE of subtrees:
 Tree   #Tnodes  Mean MSE   SE(Mean)    BSE(Mean)  Median MSE  BSE(Median)
   1       13   1.878E-01   2.882E-02   2.376E-02   1.878E-01   3.220E-02
   2       12   1.878E-01   2.882E-02   2.376E-02   1.878E-01   3.220E-02
   3       11   1.871E-01   2.883E-02   2.391E-02   1.878E-01   3.347E-02
   4       10   1.867E-01   2.885E-02   2.404E-02   1.878E-01   3.390E-02
   5        9   1.888E-01   2.901E-02   2.414E-02   1.895E-01   3.891E-02
   6        8   1.890E-01   2.901E-02   2.428E-02   1.895E-01   3.891E-02
   7        7   1.885E-01   2.887E-02   2.519E-02   1.989E-01   3.953E-02
   8        6   1.762E-01   2.739E-02   2.459E-02   1.651E-01   3.081E-02
   9        5   1.769E-01   2.714E-02   2.334E-02   1.651E-01   2.637E-02
  10        4   1.768E-01   2.180E-02   2.222E-02   1.725E-01   2.574E-02
  11**      3   1.677E-01   2.150E-02   2.285E-02   1.643E-01   3.291E-02
  12++      2   1.887E-01   2.313E-02   2.491E-02   1.733E-01   4.050E-02
  13        1   4.436E-01   3.247E-02   3.208E-02   4.574E-01   4.953E-02

0-SE tree based on mean is marked with *
0-SE tree based on median is marked with +
Selected-SE tree based on mean using naive SE is marked with **
Selected-SE tree based on mean using bootstrap SE is marked with --
Selected-SE tree based on median and bootstrap SE is marked with ++
* tree same as + tree
** tree same as + tree
** tree same as -- tree
* tree same as ** tree
* tree same as -- tree


Following tree is based on mean CV with naive SE estimate (**).

Structure of final tree. Each terminal node is marked with a T.

D-mean is mean of Logsalary in the node
Cases fit give the number of cases used to fit node
MSE and R^2 are based on all cases in node
       Node    Total    Cases Matrix    Node      Node      Node  Split         Other
```

```
      label   cases      fit  rank    D-mean     MSE       R^2  variable      variables
          1     263      263     2  5.945E+00  4.502E-01  0.4257 Yrs  +Hitcr
          2     143      143     2  5.506E+00  1.284E-01  0.8254 Hitcr  +Batcr
         4T     110      110     2  5.146E+00  9.210E-02  0.7371 Wlkcr  +Hitcr
         5T      33       33     2  6.706E+00  7.500E-02  0.4395 Wlk86  +Rbcr
         3T     120      120     2  6.469E+00  1.943E-01  0.4331 Wlkcr  +Hit86
```

The last column, labeled ‘‘Fit variables’’, give the regressor variable names and the signs of their regression coefficients.

```
Number of terminal nodes of final tree: 3
Total number of nodes of final tree:    5

Regression tree:

 Node 1: Yrs <=    6.00000 or NA
   Node 2: Hitcr <=  4.59500E+02 or NA
     Node 4: Logsalary-mean =   5.14642
   Node 2: Hitcr >  4.59500E+02 and not NA
     Node 5: Logsalary-mean =   6.70595
 Node 1: Yrs >    6.00000 and not NA
   Node 3: Logsalary-mean =   6.46866

***************************************************************


Node 1: Intermediate node
A case goes into Node 2 if Yrs <=  6.0000000E+00 or NA
Yrs mean =  7.3802E+00
Coefficients of least squares regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   5.2933E+00       84.65 0.0000
Hitcr      8.8852E-04       13.91 0.0000  4.2000E+01  7.3392E+02  4.2560E+03
Mean of Logsalary =   5.9454102795235446
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
----------------------------
Node 2: Intermediate node
A case goes into Node 4 if Hitcr <=  4.5950000E+02 or NA
Hitcr mean =  3.2022E+02
----------------------------
Node 4: Terminal node
Coefficients of least squares regression functions:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   4.2487E+00       71.82 0.0000
Hitcr      4.1935E-03       17.40 0.0000  4.2000E+01  2.1408E+02  4.5700E+02
Mean of Logsalary =   5.1464245329622393
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
```

```
----------------------------
Node 5: Terminal node
Coefficients of least squares regression functions:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   5.9484E+00       36.98 0.0000
Rbcr       2.5219E-03        4.93 0.0000  1.0300E+02  3.0039E+02  4.7500E+02
Mean of Logsalary =   6.7059525135315550
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
----------------------------
Node 3: Terminal node
Coefficients of least squares regression functions:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   5.4482E+00       47.48 0.0000
Hit86      9.4846E-03        9.50 0.0000  3.2000E+01  1.0759E+02  2.0000E+02
Mean of Logsalary =   6.4686647661858760
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
----------------------------

Proportion of variance (R-squared) explained by tree model =  .8279

Observed and fitted values are stored in lin.fit
LaTeX code for tree is in lin.tex
```

The LaTeX tree is shown in Figure 5.

5.3 ANCOVA models

Besides, multiple linear, stepwise linear, and best simple polynomial regression, GUIDE can also fit a a best ANCOVA model in each node. The ANCOVA model uses stepwise regression to find the best single linear regressor and the best subset of dummy (indicator) variables constructed from any B variables.

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice: 1
Name of batch input file: ancova.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: ancova.out
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
Choose type of regression model:
 1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
```

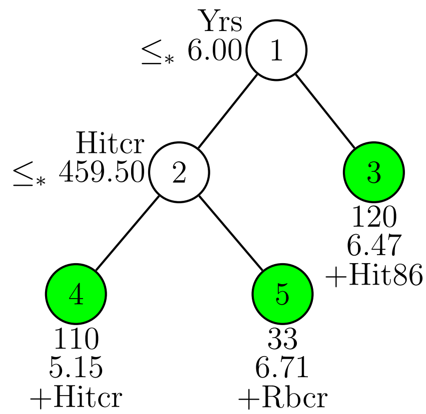

Figure 5: GUIDE piecewise simple linear least-squares regression tree for predicting Logsalary. At each intermediate node, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Sample sizes, means of Logsalary, and signs and names of regressor variable printed below nodes. Terminal nodes with negative, zero, and positive slopes are colored red, yellow, and green, respectively.

```
 5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
Input choice ([1:6], <cr>=1):
Input 1 for least squares, 2 least median of squares ([1:2], <cr>=1):
Choose complexity of model to use at each node:
0: stepwise linear, 1: multiple linear, 2: best polynomial, 3: constant,
4: stepwise simple ANCOVA ([0:4], <cr>=0): 4
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: bbdsc.txt
Reading data description file ...
Training sample file: bbdat.txt
Missing value code: NA
Records in data file start on line 1
Dependent variable is Logsalary
Reading data file ...
Number of records in data file: 263
Length of longest data entry: 17
Checking for missing values ...
Total number of cases: 263
Col. no. Categorical variable    #levels    #missing values
      16 Leag86                        2                  0
      17 Div86                         2                  0
```

```
      18 Team86                       24                 0
      19 Pos86                        23                 0
      24 Leag87                        2                 0
      25 Team87                       24                 0
Re-checking data ...
Assigning codes to categorical and missing values
firstln =           1
Finish checking data
GUIDE will try to create the variables in the description file.
If it is unsuccessful, please create the columns yourself...
Number of dummy variables created:            25
Creating dummy variables
Rereading data
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       263          0          0        3       16        0        0        4        2
No weight variable in data file
No. cases used for training: 263
Finish reading data file
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): ancova.tex
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
Input name of file to store node ID and fitted value of each case: ancova.fit
Input file is created!
Run GUIDE with the command: guide < ancova.in
```

Results

```
Least squares regression tree
Predictions truncated at global min and max of D sample values
Pruning by cross-validation
Data description file: bbdsc.txt
Training sample file: bbdat.txt
Missing value code: NA
Records in data file start on line 1
Dependent variable is Logsalary
Piecewise simple linear ANCOVA model
F-to-enter and F-to-delete =  4.000 3.990
Number of records in data file: 263
Length of longest data entry: 17
Number of dummy variables created =  25

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
```

```
 Column  Name            Minimum      Maximum   #Categories       #Missing
      3  Bat86      n   1.2700E+02   6.8700E+02
      4  Hit86      n   3.2000E+01   2.3800E+02
      5  Hr86       n   0.0000E+00   4.0000E+01
      6  Run86      n   1.3000E+01   1.3000E+02
      7  Rb86       n   8.0000E+00   1.2100E+02
      8  Wlk86      n   3.0000E+00   1.0500E+02
      9  Yrs        n   1.0000E+00   2.4000E+01
     10  Batcr      n   1.8100E+02   1.4053E+04
     11  Hitcr      n   4.2000E+01   4.2560E+03
     12  Hrcr       n   0.0000E+00   5.4800E+02
     13  Runcr      n   1.8000E+01   2.1650E+03
     14  Rbcr       n   9.0000E+00   1.6590E+03
     15  Wlkcr      n   8.0000E+00   1.5660E+03
     16  Leag86     b                                       2
     17  Div86      b                                       2
     18  Team86     c                                      24
     19  Pos86      b                                      23
     20  Puto86     n   0.0000E+00   1.3770E+03
     21  Asst86     n   0.0000E+00   4.9200E+02
     22  Err86      n   0.0000E+00   3.2000E+01
     24  Leag87     b                                       2
     25  Team87     c                                      24
     26  Logsalary  d   4.2121E+00   7.8079E+00
  =================== Constructed variables ===================
     27  Leag8.N    f   0.0000E+00   1.0000E+00
     28  Div86.W    f   0.0000E+00   1.0000E+00
     29  Pos86.1O   f   0.0000E+00   1.0000E+00
     30  Pos86.23   f   0.0000E+00   1.0000E+00
     31  Pos86.2B   f   0.0000E+00   1.0000E+00
     32  Pos86.2S   f   0.0000E+00   1.0000E+00
     33  Pos86.32   f   0.0000E+00   1.0000E+00
     34  Pos86.3B   f   0.0000E+00   1.0000E+00
     35  Pos86.3O   f   0.0000E+00   1.0000E+00
     36  Pos86.3S   f   0.0000E+00   1.0000E+00
     37  Pos86.C    f   0.0000E+00   1.0000E+00
     38  Pos86.CD   f   0.0000E+00   1.0000E+00
     39  Pos86.CF   f   0.0000E+00   1.0000E+00
     40  Pos86.DH   f   0.0000E+00   1.0000E+00
     41  Pos86.DO   f   0.0000E+00   1.0000E+00
     42  Pos86.LF   f   0.0000E+00   1.0000E+00
     43  Pos86.O1   f   0.0000E+00   1.0000E+00
     44  Pos86.OD   f   0.0000E+00   1.0000E+00
     45  Pos86.OF   f   0.0000E+00   1.0000E+00
     46  Pos86.OS   f   0.0000E+00   1.0000E+00
     47  Pos86.RF   f   0.0000E+00   1.0000E+00
```

```
     48  Pos86.S3   f   0.0000E+00   1.0000E+00
     49  Pos86.SS   f   0.0000E+00   1.0000E+00
     50  Pos86.UT   f   0.0000E+00   1.0000E+00
     51  Leag8.N    f   0.0000E+00   1.0000E+00

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       263         0          0        3       16        0        0        4        2
 No weight variable in data file
 No. cases used for training: 263

 Missing values imputed with node means for model fitting
 Interaction tests on all variables
 Pruning by v-fold cross-validation, with v =  10
 Selected tree is based on mean of CV estimates
 Fraction of cases used for splitting each node =   1.0000
 Max number of split levels =  10
 Minimum node size =  13

 Pruning sequence
   Subtree     Pruned  #Terminal        True           Geometric
    number       node      nodes        alpha             mean
         0          0         17       0.0000          0.0000
         1          9         16       0.0000          0.0000
         2         24         15       0.0000          0.0000
         3         17         14      0.28425E-02     0.28911E-02
         4         32         13      0.29406E-02     0.35528E-02
         5         16         12      0.42925E-02     0.47688E-02
         6          5         11      0.52980E-02     0.58824E-02
         7          8         10      0.65313E-02     0.89049E-02
         8         12          9      0.12141E-01     0.12154E-01
         9          6          6      0.12166E-01     0.12668E-01
        10          3          4      0.13191E-01     0.14523E-01
        11          4          3      0.15990E-01     0.18373E-01
        12          2          2      0.21112E-01     0.74707E-01
        13          1          1      0.26436         0.17977+309

 Number of SE's for pruned tree =   5.0000E-01

 Size and CV MSE and SE of subtrees:
  Tree    #Tnodes  Mean MSE   SE(Mean)    BSE(Mean)  Median MSE  BSE(Median)
    1       16   2.526E-01   2.763E-02   3.729E-02   2.140E-01   5.299E-02
    2       15   2.526E-01   2.763E-02   3.729E-02   2.140E-01   5.299E-02
    3       14   2.578E-01   2.804E-02   3.620E-02   2.218E-01   4.748E-02
    4       13   2.508E-01   2.921E-02   3.522E-02   2.218E-01   4.024E-02
    5       12   2.428E-01   2.882E-02   3.128E-02   2.212E-01   3.581E-02
```

```
   6       11   2.422E-01   2.880E-02   3.124E-02   2.201E-01   3.770E-02
   7       10   2.443E-01   3.026E-02   3.276E-02   2.201E-01   4.369E-02
   8        9   2.230E-01   2.828E-02   3.338E-02   2.025E-01   3.599E-02
   9        6   2.285E-01   2.859E-02   3.416E-02   2.025E-01   4.248E-02
  10        4   2.137E-01   2.664E-02   2.837E-02   2.021E-01   3.666E-02
  11        3   1.897E-01   2.400E-02   2.585E-02   1.959E-01   3.605E-02
  12**      2   1.845E-01   2.378E-02   2.321E-02   1.658E-01   3.923E-02
  13        1   4.259E-01   3.050E-02   3.442E-02   4.367E-01   5.145E-02

0-SE tree based on mean is marked with *
0-SE tree based on median is marked with +
Selected-SE tree based on mean using naive SE is marked with **
Selected-SE tree based on mean using bootstrap SE is marked with --
Selected-SE tree based on median and bootstrap SE is marked with ++
** tree and ++ tree are the same


Following tree is based on mean CV with naive SE estimate (**).

Structure of final tree. Each terminal node is marked with a T.

D-mean is mean of Logsalary in the node
Cases fit give the number of cases used to fit node
MSE and R^2 are based on all cases in node
       Node    Total    Cases Matrix    Node       Node      Node  Split         Other
       label   cases      fit  rank    D-mean     MSE       R^2   variable      variables
          1      263      263     5  5.945E+00  4.160E-01  0.4755 Yrs  +Hitcr
         2T      143      143     4  5.506E+00  1.184E-01  0.8413 Hitcr  +Runcr
         3T      120      120     4  6.469E+00  1.838E-01  0.4729 Wlk86  +Hit86
 The linear predictor is the one with a + sign under the ``Other variable'' column.

Number of terminal nodes of final tree: 2
Total number of nodes of final tree:    3

Regression tree:

 Node 1: Yrs <=    6.00000 or NA
   Node 2: Logsalary-mean =   5.50632
 Node 1: Yrs >    6.00000 and not NA
   Node 3: Logsalary-mean =   6.46866

***************************************************************


Node 1: Intermediate node
A case goes into Node 2 if Yrs <=  6.0000000E+00 or NA
```

```
Yrs mean =  7.3802E+00
Coefficients of least squares regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   5.3659E+00       72.24 0.0000
Hitcr      8.9484E-04       14.45 0.0000  4.2000E+01  7.3392E+02  4.2560E+03
Div86.W   -1.8665E-01       -2.34 0.0000  0.0000E+00  5.0951E-01  1.0000E+00
Pos86.RF   4.6442E-01        3.22 0.0000  0.0000E+00  8.3650E-02  1.0000E+00
Pos86.UT  -5.5260E-01       -2.63 0.0000  0.0000E+00  3.8023E-02  1.0000E+00
Mean of Logsalary =   5.9454102795235446
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
----------------------------
Node 2: Terminal node
Coefficients of least squares regression functions:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   4.4528E+00       89.49 0.0000
Runcr      6.8396E-03       26.94 0.0000  1.8000E+01  1.6197E+02  5.2900E+02
Pos86.2B  -2.8035E-01       -2.72 0.0000  0.0000E+00  9.0909E-02  1.0000E+00
Pos86.CF  -2.7430E-01       -2.89 0.0000  0.0000E+00  1.0490E-01  1.0000E+00
Mean of Logsalary =   5.5063156054013094
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
----------------------------
Node 3: Terminal node
Coefficients of least squares regression functions:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   5.5298E+00       46.67 0.0000
Hit86      9.2208E-03        9.08 0.0000  3.2000E+01  1.0759E+02  2.0000E+02
Pos86.2B  -2.7882E-01       -2.19 0.0000  0.0000E+00  1.0833E-01  1.0000E+00
Pos86.UT  -3.4458E-01       -2.11 0.0000  0.0000E+00  6.6667E-02  1.0000E+00
Mean of Logsalary =   6.4686647661858760
Predicted values truncated at   4.2121275978784798      &   7.8079166289264101
----------------------------

Proportion of variance (R-squared) explained by tree model =  .8172

Observed and fitted values are stored in ancova.fit
LaTeX code for tree is in ancova.tex
```

The LaTeX tree, shown in Figure 6.

5.4 Quantile regression

Instead of estimating the conditional mean, we can estimate conditional quantiles (Chaudhuri and Loh, 2002; Koenker and Bassett, 1978). We demonstrate this with the data set `tuitiondat.txt`, which gives information on tuition and other variables for U.S. colleges. The data description file `tuitiondsc.txt` is:

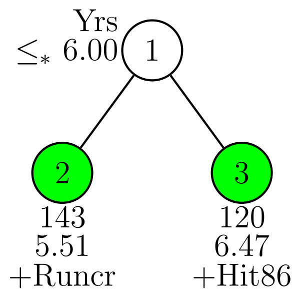

Figure 6: GUIDE piecewise ANCOVA regression tree for predicting Logsalary. At each intermediate node, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Sample sizes, means of Logsalary, and signs and names of regressor variable printed below nodes. Terminal nodes with negative, zero, and positive slopes are colored red, yellow, and green, respectively.

```
tuitiondat.dat
NA
1
1  FICE x
2  CollName x
3  State x
4  PubPriv c
5  MathSAT x
6  VerbSAT x
7  CombSAT n
8  ACT x
9  Q1MSAT x
10 Q3MSAT x
11 Q1VSAT x
12 Q3VSAT x
13 Q1ACT x
14 Q3ACT x
15 AppsRec n
16 AppsAcc n
17 NewEnrol n
18 Top10 n
19 Top25 n
20 FUgrad n
21 PUgrad x
22 InTuition x
23 OutTuition d
24 RnBcost n
```

```
25 RmCost x
26 BrdCost x
27 AddFees x
28 BookCost x
29 PerSpend x
30 PFacPhD n
31 PFacTerm x
32 StudFac n
33 PAlDonate x
34 InstExp n
35 GradRate n
36 Type c
37 FullPSal n
38 AssocPSal x
39 AsstPSal x
40 AveSal x
41 FullPComp x
42 AssocPComp x
43 AsstPComp x
44 AveComp x
45 NFullProf n
46 NAssocProf x
47 NAsstProf x
48 NInstr x
49 NAllFac x
```

Following is a session log to create an input file for constructing a piecewise simple linear tree for the 90th percentile of out-of-state tuition (`OutTuition`).

```
 0. Read the warranty disclaimer
 1. Create an input file for batch run
 2. Fit a model without creating input file
 3. Convert data to other formats
 4. Variable importance scoring and differential item functioning
 Input your choice: 1
 Name of batch input file: quant.in
 Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
 Name of batch output file: quant.out
 Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
 Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
 Choose type of regression model:
  1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
  5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
 Input choice ([1:6], <cr>=1): 2
 Choose complexity of model to use at each node:
```

```
1: multiple linear, 2: best polynomial, 3: constant ([1:3], <cr>=1): 2
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):
Input 1 for 1 quantile, 2 for 2 quantiles ([1:2], <cr>=1):
 Option 2 allows simultaneous modeling of a pair of quantile values (e.g., 0.1 and 0.9)
Input quantile probability ([0.00:1.00], <cr>=0.50): 0.9

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: tuitiondsc.txt
Reading data description file ...
Training sample file: tuitiondat.txt
Missing value code: NA
Records in data file start on line 1
Warning: B variables changed to C
Dependent variable is OutTuition
Reading data file ...
Number of records in data file: 1134
Length of longest data entry: 20
Checking for missing values ...
Total number of cases: 1134
Col. no. Categorical variable    #levels    #missing values
       4 PubPriv                       2                  0
      36 Type                          3                  0
Re-checking data ...
Allocating missing value information
Assigning codes to categorical and missing values
Finish checking data
Creating missing value indicators
Rereading data
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
      1134         13        621       32       14        0        0        0        2
No. cases used for training: 1121
No. cases excluded due to 0 weight or missing D: 13
Finish reading data file
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): quant.tex
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
Input name of file to store node ID and fitted value of each case: quant.fit
Input file is created!
Run GUIDE with the command: guide < quant.in
```

Results

```
Quantile regression tree with quantile probability .9000
No truncation of predicted values
```

```
Pruning by cross-validation
Data description file: tuitiondsc.txt
Training sample file: tuitiondat.txt
Missing value code: NA
Records in data file start on line 1
Warning: B variables changed to C
Dependent variable is OutTuition
Piecewise simple linear or constant model
Number of records in data file: 1134
Length of longest data entry: 20

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
 Column  Name            Minimum      Maximum   #Categories       #Missing
      4  PubPriv     c                                    2
      7  CombSAT     n   6.0000E+02   1.4100E+03                       471
     15  AppsRec     n   5.7000E+01   4.8094E+04                         9
     16  AppsAcc     n   4.4000E+01   2.6330E+04                         9
     17  NewEnrol    n   2.1000E+01   7.4250E+03                         5
     18  Top10       n   1.0000E+00   9.8000E+01                       183
     19  Top25       n   1.1000E+01   1.0000E+02                       155
     20  FUgrad      n   1.1800E+02   3.1643E+04                         3
     23  OutTuition  d   1.0440E+03   2.5750E+04                        13
     24  RnBcost     n   1.3060E+03   8.7000E+03                        57
     30  PFacPhD     n   8.0000E+00   1.0500E+02                        29
     32  StudFac     n   2.5000E+00   4.2600E+01                         2
     34  InstExp     n   1.8340E+03   6.2469E+04                        24
     35  GradRate    n   8.0000E+00   1.1800E+02                        69
     36  Type        c                                    3
     37  FullPSal    n   2.7000E+02   1.0090E+03                        61
     45  NFullProf   n   0.0000E+00   9.9700E+02

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
      1134         13        621       32       14        0        0        0        2
No. cases used for training: 1121
No. cases excluded due to 0 weight or missing D: 13

Missing values imputed with node means for model fitting
Interaction tests on all variables
Pruning by v-fold cross-validation, with v =  10
Selected tree is based on mean of CV estimates
Fraction of cases used for splitting each node =   1.0000
Max number of split levels =  11
Minimum node size =  55
```

```
 Pruning sequence
   Subtree     Pruned  #Terminal        True           Geometric
    number       node      nodes        alpha             mean
         0          0         14       0.0000          0.0000
         1         10         13       1.1002          1.3727
         2         13         12       1.7128          2.4568
         3          4         11       3.5241          3.6666
         4         25         10       3.8150          4.4876
         5         14          9       5.2788          5.3455
         6         12          8       5.4132          6.7830
         7          2          6       8.4995          9.3094
         8         15          5       10.196          10.879
         9          6          4       11.608          14.586
        10          7          3       18.329          24.342
        11          3          2       32.328          51.239
        12          1          1       81.213         0.17977+309

 Number of SE's for pruned tree =   5.0000E-01

 Size and CV Loss and SE of subtrees:
  Tree   #Tnodes  Mean Loss   SE(Mean)   BSE(Mean)  Median Loss  BSE(Median)
    1       13   3.523E+02   1.516E+01   1.322E+01   3.463E+02   1.016E+01
    2       12   3.486E+02   1.498E+01   1.359E+01   3.454E+02   1.305E+01
    3       11   3.494E+02   1.501E+01   1.484E+01   3.454E+02   1.407E+01
    4+      10   3.506E+02   1.495E+01   1.466E+01   3.454E+02   1.484E+01
    5        9   3.531E+02   1.493E+01   1.460E+01   3.509E+02   1.625E+01
    6        8   3.483E+02   1.460E+01   1.439E+01   3.469E+02   1.552E+01
    7        6   3.495E+02   1.437E+01   1.416E+01   3.509E+02   1.593E+01
    8*       5   3.452E+02   1.435E+01   1.213E+01   3.509E+02   1.623E+01
    9**      4   3.494E+02   1.445E+01   1.100E+01   3.509E+02   1.410E+01
   10        3   3.711E+02   1.457E+01   1.424E+01   3.641E+02   1.237E+01
   11        2   3.837E+02   1.356E+01   6.964E+00   3.872E+02   8.182E+00
   12        1   4.636E+02   1.426E+01   1.137E+01   4.569E+02   2.149E+01

 0-SE tree based on mean is marked with *
 0-SE tree based on median is marked with +
 Selected-SE tree based on mean using naive SE is marked with **
 Selected-SE tree based on mean using bootstrap SE is marked with --
 Selected-SE tree based on median and bootstrap SE is marked with ++
 ** tree same as ++ tree
 ** tree same as -- tree
 ++ tree same as -- tree


 Following tree is based on mean CV with naive SE estimate (**).
```

```
Structure of final tree. Each terminal node is marked with a T.

D-quant is quantile of OutTuition in the node
Cases fit give the number of cases used to fit node
       Node    Total    Cases Matrix    Node    Split          Other
       label   cases      fit  rank    D-quant  variable       variables
          1     1121     1098     2  1.591E+04  PubPriv
          2T     434      432     2  8.820E+03  RnBcost
          3      687      672     2  1.745E+04  InstExp
          6T     421      406     2  1.250E+04  RnBcost
          7      266      261     2  1.903E+04  RnBcost
         14T     149      143     2  1.708E+04  InstExp
         15T     117      113     2  1.951E+04  Top10

Number of terminal nodes of final tree: 4
Total number of nodes of final tree:    7

Regression tree:

 Node 1: PubPriv = "Public"
   Node 2: OutTuition sample quantile =  8.82000E+03
 Node 1: PubPriv /= "Public"
   Node 3: InstExp <=  9.55500E+03 or NA
     Node 6: OutTuition sample quantile =  1.25000E+04
   Node 3: InstExp >  9.55500E+03 and not NA
     Node 7: RnBcost <=  5.55750E+03
       Node 14: OutTuition sample quantile =  1.70800E+04
     Node 7: RnBcost >  5.55750E+03 or NA
       Node 15: OutTuition sample quantile =  1.95100E+04

***************************************************************

In the following the predictor node mean is mean of complete cases
Regression coefficients are computed from the complete cases

Node 1: Intermediate node
A case goes into Node 2 if PubPriv = "Public"
PubPriv mode = "Private"
Coefficients of quantile regression function:
Regressor  Coefficient     Minimum        Mean     Maximum
Constant    3.7443E+03
InstExp     1.0271E+00  1.8340E+03  9.0272E+03  6.2469E+04
----------------------------
Node 2: Terminal node
Coefficients of quantile regression function:
```

```
Regressor  Coefficient     Minimum        Mean     Maximum
Constant   -8.9699E+02
FullPSal    1.6871E+01  3.5900E+02  5.4766E+02  8.9300E+02
----------------------------
Node 3: Intermediate node
A case goes into Node 6 if InstExp <=  9.5550000E+03 or NA
InstExp mean =  1.0340E+04
----------------------------
Node 6: Terminal node
Coefficients of quantile regression function:
Regressor  Coefficient     Minimum        Mean     Maximum
Constant    6.0532E+03
InstExp     7.9332E-01  2.5890E+03  7.3261E+03  9.5530E+03
----------------------------
Node 7: Intermediate node
A case goes into Node 14 if RnBcost <=  5.5575000E+03
RnBcost mean =  5.2677E+03
----------------------------
Node 14: Terminal node
Coefficients of quantile regression function:
Regressor  Coefficient     Minimum        Mean     Maximum
Constant    6.4973E+03
FullPSal    1.8643E+01  3.0700E+02  5.4524E+02  9.7000E+02
----------------------------
Node 15: Terminal node
Coefficients of quantile regression function:
Regressor  Coefficient     Minimum        Mean     Maximum
Constant    2.3847E+04
RnBcost    -7.2884E-01  5.5650E+03  6.2796E+03  8.1240E+03
----------------------------

Observed and fitted values are stored in quant.fit
LaTeX code for tree is in quant.tex
```

The LaTeX tree is shown in Figure 7 and plots of the data in the terminal nodes of the tree are given in Figure 8.

5.5 Least median of squares

Although median regression may be preferred to least-squares regression if there are large outliers in a data set, an alternative that is even more robust to outliers is *least median of squares* regression (Rousseeuw and Leroy, 1987). GUIDE can construct tree models using this criterion. We use the college tuition data for illustration. A

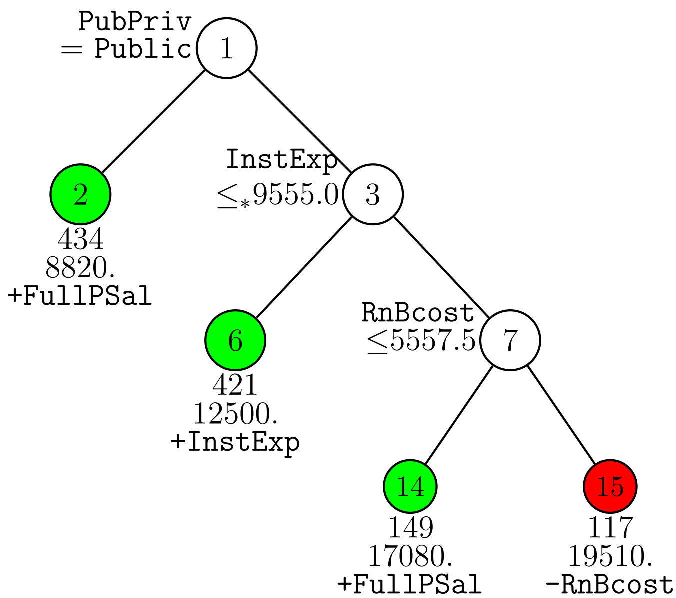

Figure 7: GUIDE 0.50-SE piecewise simple linear .900-quantile regression tree for predicting `OutTuition`. At each split, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Sample sizes, 0.90-quantiles of `OutTuition`, and signs and names of best regressor are printed below nodes. Terminal nodes with negative, zero, and positive slopes are colored red, yellow, and green, respectively.

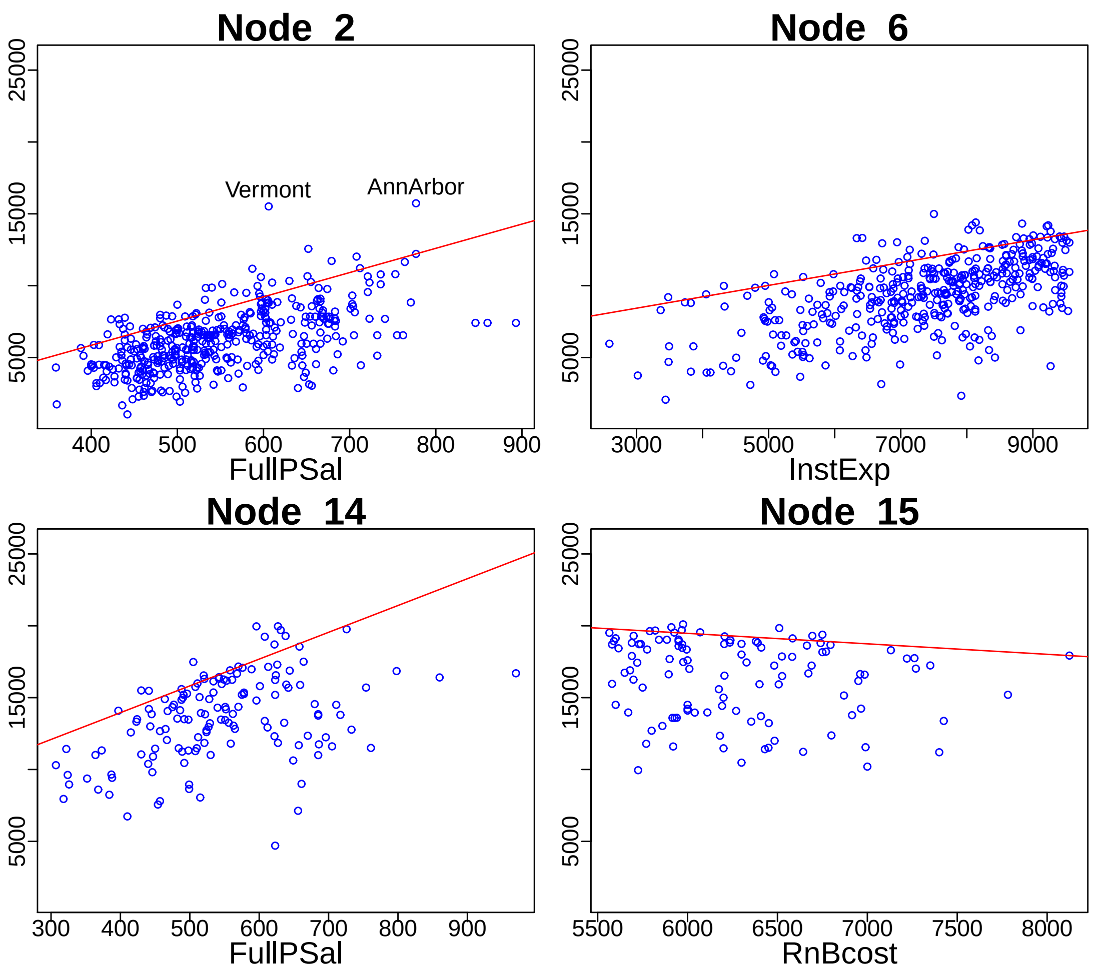

Figure 8: Plots of data and estimated linear 0.9-quantile fits in terminal nodes of tree in Figure 7.

session log of the input file generation is below, followed by the results and the LaTeX tree diagram in Figure 9.

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice: 1
Name of batch input file: lms.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: lms.out
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1):
Choose type of regression model:
 1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
 5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
Input choice ([1:6], <cr>=1):
Input 1 for least squares, 2 least median of squares ([1:2], <cr>=1): 2
 This is where the option for least median of squares is selected.
Choose complexity of model to use at each node:
1: multiple linear, 2: best simple linear, 3: constant ([1:3], <cr>=2):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

Input name of data description file (maximum 100 characters; enclose within quotes
if it contains spaces or non alphanumeric characters): tuitiondsc.txt
Reading data description file ...
Training sample file: tuitiondat.txt
Missing value code: NA
Records in data file start on line 1
Warning: B variables changed to C
Dependent variable is OutTuition
Reading data file ...
Number of records in data file: 1134
Length of longest data entry: 20
Checking for missing values ...
Total number of cases: 1134
Col. no. Categorical variable    #levels    #missing values
       4 PubPriv                       2                  0
      36 Type                          3                  0
Re-checking data ...
Allocating missing value information
Assigning codes to categorical and missing values
Finish checking data
Creating missing value indicators
Rereading data
```

```
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
      1134         13        621       32       14        0        0        0        2
No weight variable in data file
No. cases used for training: 1121
No. cases excluded due to 0 weight or missing D: 13
Finish reading data file
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): lms.tex
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
Input name of file to store node ID and fitted value of each case: lms.fit
Input file is created!
Run GUIDE with the command: guide < lms.in
```

Results

```
Least median of squares regression tree
Predictions truncated at global min and max of D sample values
Pruning by cross-validation
Data description file: tuitiondsc.txt
Training sample file: tuitiondat.txt
Missing value code: NA
Records in data file start on line 1
Warning: B variables changed to C
Dependent variable is OutTuition
Piecewise simple linear or constant model
Number of records in data file: 1134
Length of longest data entry: 20

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
 Column  Name            Minimum      Maximum   #Categories       #Missing
      4  PubPriv     c                                    2
      7  CombSAT     n   6.0000E+02   1.4100E+03                       471
     15  AppsRec     n   5.7000E+01   4.8094E+04                         9
     16  AppsAcc     n   4.4000E+01   2.6330E+04                         9
     17  NewEnrol    n   2.1000E+01   7.4250E+03                         5
     18  Top10       n   1.0000E+00   9.8000E+01                       183
     19  Top25       n   1.1000E+01   1.0000E+02                       155
     20  FUgrad      n   1.1800E+02   3.1643E+04                         3
     23  OutTuition  d   1.0440E+03   2.5750E+04                        13
     24  RnBcost     n   1.3060E+03   8.7000E+03                        57
     30  PFacPhD     n   8.0000E+00   1.0500E+02                        29
     32  StudFac     n   2.5000E+00   4.2600E+01                         2
```

```
     34  InstExp     n   1.8340E+03   6.2469E+04                        24
     35  GradRate    n   8.0000E+00   1.1800E+02                        69
     36  Type        c                                     3
     37  FullPSal    n   2.7000E+02   1.0090E+03                        61
     45  NFullProf   n   0.0000E+00   9.9700E+02

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
      1134        13        621       32       14        0        0        0        2
No weight variable in data file
No. cases used for training: 1121
No. cases excluded due to 0 weight or missing D: 13

Missing values imputed with node means for model fitting
Interaction tests on all variables
Pruning by v-fold cross-validation, with v =  10
Selected tree is based on mean of CV estimates
Fraction of cases used for splitting each node =   1.0000
Max number of split levels =  11
Minimum node size =  55

Pruning sequence
  Subtree     Pruned  #Terminal        True           Geometric
   number       node      nodes        alpha             mean
        0          0         10       0.0000          0.0000
        1          4          9       12.053          12.211
        2         27          8       12.371          13.817
        3          5          6       15.433          25.588
        4          2          5       42.424          50.631
        5         13          4       60.424          68.739
        6          6          3       78.199          107.01
        7          3          2       146.44          241.25
        8          1          1       397.44         0.17977+309

Number of SE's for pruned tree =   5.0000E-01

Size and CV median absolute residual (MAR) and SE of subtrees:
 Tree   #Tnodes  Mean MAR   BSE(Mean)  Median MAR  BSE(Median)
   1--       9   1.362E+05   4.854E+01   1.187E+03   7.998E+01
   2++       8   1.363E+05   4.888E+01   1.187E+03   8.021E+01
   3         6   1.386E+05   4.543E+01   1.236E+03   6.459E+01
   4         5   1.457E+05   2.781E+01   1.322E+03   5.666E+01
   5         4   1.425E+05   3.382E+01   1.298E+03   5.667E+01
   6         3   1.465E+05   5.338E+01   1.236E+03   7.971E+01
   7         2   1.688E+05   8.907E+01   1.535E+03   1.376E+02
   8         1   1.901E+05   8.738E+01   1.716E+03   9.536E+01
```

```
The selected tree is marked by two dashes.
0-SE tree based on mean is marked with *
0-SE tree based on median is marked with +
Selected-SE tree based on mean using bootstrap SE is marked with --
Selected-SE tree based on median and bootstrap SE is marked with ++
+ tree same as ++ tree
* tree same as -- tree

Following tree is based on mean CV with bootstrap SE estimate (--).

Structure of final tree. Each terminal node is marked with a T.

D-mean is mean of OutTuition in the node
Cases fit give the number of cases used to fit node
MAR is median of absolute residuals
       Node    Total    Cases Matrix    Node      Node    Split          Other
       label   cases      fit  rank   D-median    MAR     variable       variables
          1     1121     1121     2  8.820E+03  1.687E+03 PubPriv  +InstExp
          2      434      434     2  6.114E+03  9.854E+02 NFullProf  +RnBcost
          4T     198      198     2  5.130E+03  7.377E+02 StudFac  +RnBcost
          5      236      236     2  6.832E+03  9.981E+02 InstExp  +FullPSal
         10      165      165     2  6.432E+03  9.909E+02 NewEnrol  +AppsRec
         20T      63       63     2  6.312E+03  8.499E+02 -  +FullPSal
         21T     102      102     2  6.520E+03  8.313E+02 -  +AppsRec
         11T      71       71     2  8.149E+03  9.272E+02 -  +FullPSal
          3      687      687     2  1.096E+04  1.485E+03 InstExp  +InstExp
          6      577      577     2  1.042E+04  1.314E+03 RnBcost  +InstExp
         12T     224      224     2  8.620E+03  1.239E+03 FullPSal  +InstExp
         13      353      353     2  1.143E+04  1.119E+03 AppsRec  +InstExp
         26T     145      145     2  1.030E+04  8.433E+02 RnBcost  +InstExp
         27      208      208     2  1.251E+04  9.932E+02 GradRate  +InstExp
         54T      80       80     2  1.148E+04  8.919E+02 -  +InstExp
         55T     128      128     2  1.297E+04  9.616E+02 Top10  +InstExp
          7T     110      110     2  1.788E+04  8.988E+02 -  +GradRate

Number of terminal nodes of final tree: 9
Total number of nodes of final tree:    17

Regression tree:

 Node 1: PubPriv = "Public"
   Node 2: NFullProf <=   93.00000
     Node 4: OutTuition-mean =  5.13000E+03
   Node 2: NFullProf >   93.00000 or NA
     Node 5: InstExp <=  8.58150E+03 or NA
       Node 10: NewEnrol <=  1.44850E+03
```

```
        Node 20: OutTuition-mean =  6.31200E+03
      Node 10: NewEnrol >  1.44850E+03 or NA
        Node 21: OutTuition-mean =  6.52000E+03
    Node 5: InstExp >  8.58150E+03 and not NA
      Node 11: OutTuition-mean =  8.14900E+03
Node 1: PubPriv /= "Public"
  Node 3: InstExp <=  1.38525E+04 or NA
    Node 6: RnBcost <=  3.99950E+03 or NA
      Node 12: OutTuition-mean =  8.62000E+03
    Node 6: RnBcost >  3.99950E+03 and not NA
      Node 13: AppsRec <=  9.49000E+02
        Node 26: OutTuition-mean =  1.03000E+04
      Node 13: AppsRec >  9.49000E+02 or NA
        Node 27: GradRate <=   68.00000 or NA
          Node 54: OutTuition-mean =  1.14750E+04
        Node 27: GradRate >   68.00000 and not NA
          Node 55: OutTuition-mean =  1.29700E+04
  Node 3: InstExp >  1.38525E+04 and not NA
    Node 7: OutTuition-mean =  1.78825E+04

***************************************************************

In the following the predictor node mean is mean of complete cases
Regression coefficients are computed from the complete cases

Node 1: Intermediate node
A case goes into Node 2 if PubPriv = "Public"
PubPriv mode = "Private"
Coefficients of least median of squares regression function:
Regressor  Coefficient     Minimum       Mean     Maximum
Constant      973.9978
InstExp     1.0571E+00   1834.00*******  6.2469E+04
Mean of OutTuition =   8820.0000000000000
Predicted values truncated at   1044.0000000000000     &   25750.000000000000
----------------------------
Node 2: Intermediate node
A case goes into Node 4 if NFullProf <=  9.3000000E+01
NFullProf mean =  1.7104E+02
----------------------------
Node 4: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient     Minimum       Mean     Maximum
Constant    1.3867E+03
RnBcost     1.2073E+00   1306.00*******  5.5440E+03
Mean of OutTuition =   5130.0000000000000
Predicted values truncated at   1044.0000000000000     &   25750.000000000000
```

```
----------------------------
Node 5: Intermediate node
A case goes into Node 10 if InstExp <=  8.5815000E+03 or NA
InstExp mean =  7.9108E+03
----------------------------
Node 10: Intermediate node
A case goes into Node 20 if NewEnrol <=  1.4485000E+03
NewEnrol mean =  1.8241E+03
----------------------------
Node 20: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient      Minimum        Mean     Maximum
Constant      403.6457
FullPSal    1.0536E+01    446.00*******  6.7600E+02
Mean of OutTuition =   6312.0000000000000
Predicted values truncated at   1044.0000000000000      &   25750.000000000000
----------------------------
Node 21: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient      Minimum        Mean     Maximum
Constant    3.7727E+03
AppsRec     4.2086E-01   2397.00*******  1.4939E+04
Mean of OutTuition =   6520.0000000000000
Predicted values truncated at   1044.0000000000000      &   25750.000000000000
----------------------------
Node 11: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient      Minimum        Mean     Maximum
Constant     -411.0535
FullPSal    1.3037E+01    460.00*******  8.9300E+02
Mean of OutTuition =   8149.0000000000000
Predicted values truncated at   1044.0000000000000      &   25750.000000000000
----------------------------
Node 3: Intermediate node
A case goes into Node 6 if InstExp <=  1.3852500E+04 or NA
InstExp mean =  1.0340E+04
----------------------------
Node 6: Intermediate node
A case goes into Node 12 if RnBcost <=  3.9995000E+03 or NA
RnBcost mean =  4.3413E+03
----------------------------
Node 12: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient      Minimum        Mean     Maximum
Constant     -416.4918
InstExp     1.2251E+00   3019.00*******  1.2957E+04
```

```
Mean of OutTuition =   8620.0000000000000
Predicted values truncated at   1044.0000000000000     &   25750.000000000000
----------------------------
Node 13: Intermediate node
A case goes into Node 26 if AppsRec <=  9.4900000E+02
AppsRec mean =  1.6166E+03
----------------------------
Node 26: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient      Minimum        Mean     Maximum
Constant    7.1302E+03
InstExp     3.5302E-01   3480.00*******  1.3844E+04
Mean of OutTuition =   10300.000000000000
Predicted values truncated at   1044.0000000000000     &   25750.000000000000
----------------------------
Node 27: Intermediate node
A case goes into Node 54 if GradRate <=  6.8000000E+01 or NA
GradRate mean =  7.2310E+01
----------------------------
Node 54: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient      Minimum        Mean     Maximum
Constant    2.3691E+03
InstExp     9.7252E-01   4333.00*******  1.3706E+04
Mean of OutTuition =   11475.000000000000
Predicted values truncated at   1044.0000000000000     &   25750.000000000000
----------------------------
Node 55: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient      Minimum        Mean     Maximum
Constant    6.6040E+03
InstExp     6.3919E-01   2589.00*******  1.3675E+04
Mean of OutTuition =   12970.000000000000
Predicted values truncated at   1044.0000000000000     &   25750.000000000000
----------------------------
Node 7: Terminal node
Coefficients of least median of squares regression function:
Regressor  Coefficient      Minimum        Mean     Maximum
Constant    1.3307E+04
GradRate    6.1667E+01     46.0082.1468  1.0000E+02
Mean of OutTuition =   17882.500000000000
Predicted values truncated at   1044.0000000000000     &   25750.000000000000
----------------------------

Proportion of deviance explained by tree model =   0.65999996146333617
```

```
Observed and fitted values are stored in lms.fit
LaTeX code for tree is in lms.tex
```

5.6 Poisson regression with offset

We use a data set from www.statsci.org/data/general/motorins.html on motor insurance claims in Sweden for the year 1977 (Andrews and Herzberg, 1985; Hallin and Ingenbleek, 1983). The description and data files are swedendsc.txt and swedendat.txt. The dependent variable is the number of claims. The other variables are mileagegp with ordered values 1–5, zone with 7 unordered values, bonus which is the number of years plus one since last claim, make of car with 9 unordered values, insured which is the number of insured in policy-years, and payment which is the total value of payments in Skr. We ignore insured and payments by giving them the x designation. To fit a Poisson regression model for the claim rate, we created an offset variable lninsured which is the log of insured and designate as z. Because mileagegp is an ordered categorical variable, we designate it as s to prevent it from being used as a linear predictor for fitting the Poisson node models. We designate bonus as f so that it is only used as a linear predictor and not for splitting the nodes.

```
swedendat.txt
?
1
1 mileagegp s
2 zone c
3 bonus f
4 make c
5 insured x
6 lninsured z
7 claims d
8 payments x
```

Since there is only one linear predictor, the multiple linear Poisson model is the same as the best simple linear Poisson model.

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice: 1
Name of batch input file: sweden.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: sweden.out
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
```

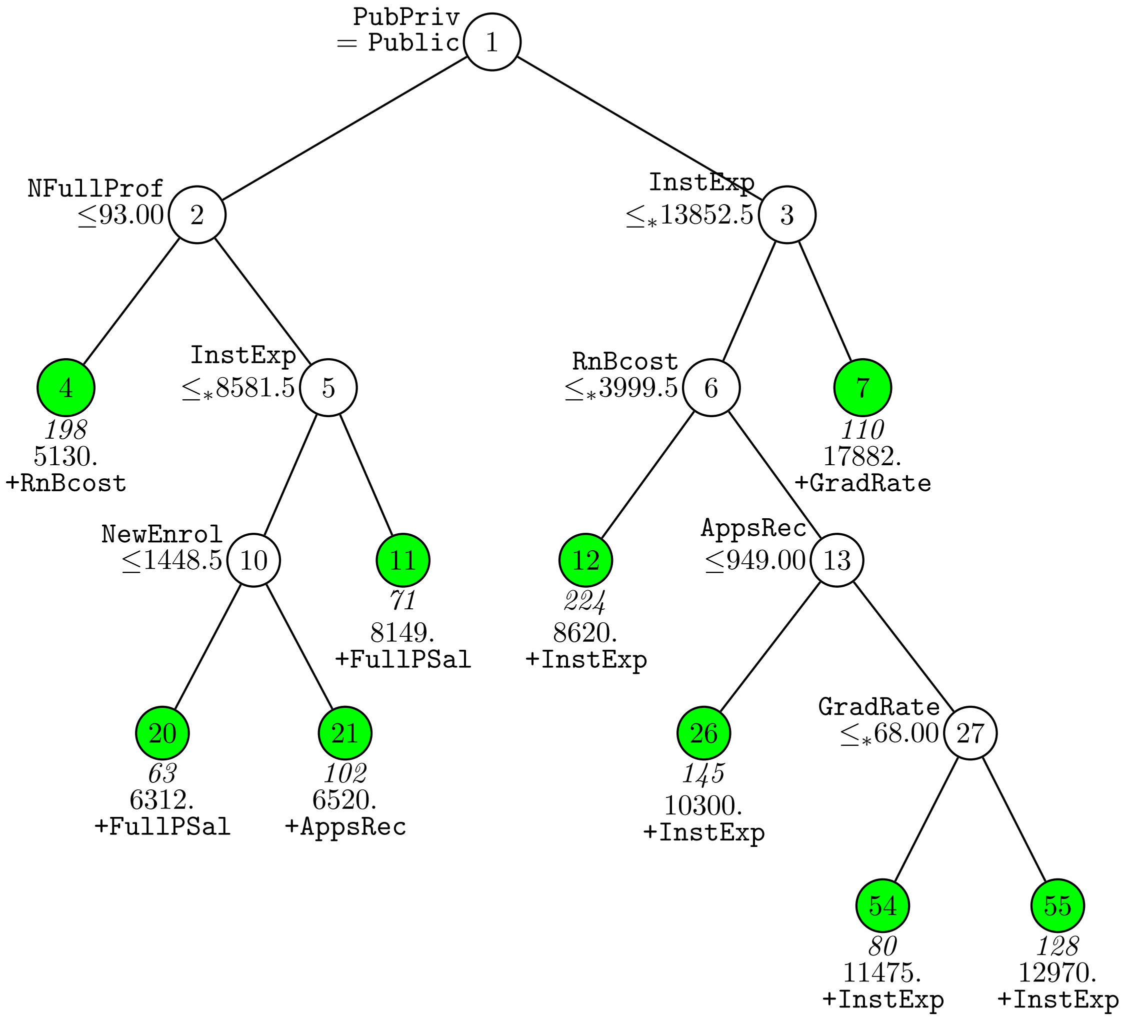

Figure 9: GUIDE 0.50-SE piecewise simple linear least-median-of-squares regression tree for predicting `OutTuition`. At each split, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Sample sizes, means of `OutTuition`, and signs and names of regressor variable are printed below nodes. Terminal nodes with negative, zero, and positive slopes are colored red, yellow, and green, respectively.

```
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
Choose type of regression model:
 1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
 5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
Input choice ([1:6], <cr>=1): 3
Choose complexity of model to use at each node:
1: multiple linear, 2: best polynomial, 3: constant ([1:3], <cr>=1):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: swedendsc.txt
Reading data description file ...
Training sample file: swedendat.txt
Missing value code: ?
Records in data file start on line 1
Dependent variable is claims
Reading data file ...
Number of records in data file: 2182
Length of longest data entry: 19
Checking for missing values ...
Total number of cases: 2182
Col. no. Categorical variable    #levels    #missing values
       2 zone                          7                  0
       4 make                          9                  0
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Number of cases with positive D values: 1797
Rereading data
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
      2182          0          0        2        0        1        1        0        2
Offset variable in column:           6
No. cases used for training: 2182
Finish reading data file
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): sweden.tex
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
Input name of file to store node ID and fitted value of each case: sweden.fit
Input file is created!
Run GUIDE with the command: guide < sweden.in
```

Results

```
Poisson regression tree
```

```
No truncation of predicted values
Pruning by cross-validation
Data description file: swedendsc.txt
Training sample file: swedendat.txt
Missing value code: ?
Records in data file start on line 1
Dependent variable is claims
Piecewise linear model
Number of records in data file: 2182
Length of longest data entry: 19
Number of cases with positive D values: 1797

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
z=offset variable
 Column  Name            Minimum      Maximum   #Categories       #Missing
      1  mileagegp  s   1.0000E+00   5.0000E+00
      2  zone       c                                     7
      3  bonus      f   1.0000E+00   7.0000E+00
      4  make       c                                     9
      6  lninsured  z  -4.6052E+00   1.1757E+01
      7  claims     d   0.0000E+00   3.3380E+03

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
      2182          0          0        2        0        1        1        0        2
Offset variable in column  6
No. cases used for training: 2182

Missing values imputed with node means for model fitting
Interaction tests on all variables
Pruning by v-fold cross-validation, with v =  10
Selected tree is based on mean of CV estimates
Fraction of cases used for splitting each node =   1.0000
Max number of split levels =  12
Minimum node size =  109

Pruning sequence
  Subtree     Pruned  #Terminal        True           Geometric
   number       node      nodes        alpha             mean
        0          0         11       0.0000          0.0000
        1         31         10      0.23796E-01     0.33717E-01
        2         61          9      0.47773E-01     0.63426E-01
        3          2          7      0.84209E-01     0.11131
        4         30          6      0.14712         0.16968
```

```
       5        15          5      0.19569        0.23210
       6         6          4      0.27528        0.40064
       7         7          3      0.58309        0.83359
       8         1          1       1.1917        0.17977+309

 Number of SE's for pruned tree =   5.0000E-01

 Size and CV Loss and SE of subtrees:
  Tree   #Tnodes  Mean Loss   SE(Mean)   BSE(Mean)  Median Loss  BSE(Median)
    1*      10   3.217E+00   2.507E-01   2.542E-01   2.870E+00   2.166E-01
    2        9   3.229E+00   2.510E-01   2.538E-01   2.964E+00   2.098E-01
    3**      7   3.301E+00   2.591E-01   2.974E-01   2.964E+00   2.109E-01
    4        6   3.458E+00   2.812E-01   2.940E-01   3.196E+00   2.070E-01
    5        5   3.584E+00   2.871E-01   2.760E-01   3.355E+00   2.192E-01
    6        4   3.566E+00   2.974E-01   2.813E-01   3.523E+00   2.159E-01
    7        3   4.243E+00   3.823E-01   3.722E-01   4.207E+00   5.342E-01
    8        1   6.519E+00   6.365E-01   5.622E-01   6.609E+00   7.858E-01

 0-SE tree based on mean is marked with *
 0-SE tree based on median is marked with +
 Selected-SE tree based on mean using naive SE is marked with **
 Selected-SE tree based on mean using bootstrap SE is marked with --
 Selected-SE tree based on median and bootstrap SE is marked with ++
 * tree same as + tree
 ** tree same as ++ tree
 ** tree same as -- tree
 ++ tree same as -- tree


 Following tree is based on mean CV with naive SE estimate (**).

 Structure of final tree. Each terminal node is marked with a T.

 Rate is mean of Y/exp(offset)
 Cases fit give the number of cases used to fit node
 Deviance is mean residual deviance for all cases in node
        Node    Total    Cases Matrix    Node       Node     Split          Other
        label   cases      fit  rank     rate     deviance   variable       variables
           1     2182     2182     2  4.749E-02  6.408E+00  make
           2T     482      482     2  3.453E-02  2.253E+00  zone
           3     1700     1700     2  4.864E-02  6.663E+00  zone
           6      490      490     2  6.375E-02  6.030E+00  zone
          12T     245      245     2  5.627E-02  4.450E+00  -
          13T     245      245     2  7.258E-02  5.188E+00  -
           7     1210     1210     2  4.209E-02  3.927E+00  mileagegp
          14T     243      243     2  3.613E-02  4.163E+00  -
```

```
         15      967      967     2  4.497E-02  2.558E+00  mileagegp
         30      490      490     2  4.368E-02  2.986E+00  zone
         60T     196      196     2  5.011E-02  2.371E+00  -
         61T     294      294     2  4.056E-02  2.316E+00  make
         31T     477      477     2  5.024E-02  1.229E+00  mileagegp

Number of terminal nodes of final tree: 7
Total number of nodes of final tree:    13

Regression tree:

 Node 1: make = "4", "6"
   Node 2: claims sample rate =  3.45272E-02
 Node 1: make /= "4", "6"
   Node 3: zone = "1", "2"
     Node 6: zone = "2"
       Node 12: claims sample rate =  5.62714E-02
     Node 6: zone /= "2"
       Node 13: claims sample rate =  7.25828E-02
   Node 3: zone /= "1", "2"
     Node 7: mileagegp <=    1.00000
       Node 14: claims sample rate =  3.61274E-02
     Node 7: mileagegp >    1.00000 or ?
       Node 15: mileagegp <=    3.00000 or ?
         Node 30: zone = "3", "5"
           Node 60: claims sample rate =  5.01083E-02
         Node 30: zone /= "3", "5"
           Node 61: claims sample rate =  4.05632E-02
       Node 15: mileagegp >    3.00000 and not ?
         Node 31: claims sample rate =  5.02372E-02

***************************************************************


Node 1: Intermediate node
A case goes into Node 2 if make =
 "4", "6"
make mode = "1"
Coefficients of loglinear regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -2.0572E+00     -305.02 0.0000
bonus     -1.8651E-01     -148.88 0.0000    1.0000E+00  4.0151E+00  7.0000E+00
Node mean for offset variable =     4.4928E+00
----------------------------
Node 2: Terminal node
Coefficients of loglinear regression function:
```

```
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -2.5849E+00     -104.45 0.0000
bonus     -1.6243E-01      -32.72 0.0000    1.0000E+00  4.0249E+00  7.0000E+00
Node mean for offset variable =     4.3366E+00
----------------------------
Node 3: Intermediate node
A case goes into Node 6 if zone =
 "1", "2"
zone mode = "1"
----------------------------
Node 6: Intermediate node
A case goes into Node 12 if zone = "2"
zone mode = "1"
----------------------------
Node 12: Terminal node
Coefficients of loglinear regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -1.8885E+00     -117.98 0.0000
bonus     -1.8707E-01      -62.66 0.0000    1.0000E+00  4.0000E+00  7.0000E+00
Node mean for offset variable =     5.1038E+00
----------------------------
Node 13: Terminal node
Coefficients of loglinear regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -1.5438E+00     -107.39 0.0000
bonus     -2.1116E-01      -75.60 0.0000    1.0000E+00  4.0000E+00  7.0000E+00
Node mean for offset variable =     4.9811E+00
----------------------------
Node 7: Intermediate node
A case goes into Node 14 if mileagegp <=  1.0000000E+00
mileagegp mean =  2.9860E+00
----------------------------
Node 14: Terminal node
Coefficients of loglinear regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -2.3934E+00     -152.64 0.0000
bonus     -1.8117E-01      -60.43 0.0000    1.0000E+00  4.0123E+00  7.0000E+00
Node mean for offset variable =     4.6679E+00
----------------------------
Node 15: Intermediate node
A case goes into Node 30 if mileagegp <=  3.0000000E+00 or ?
mileagegp mean =  3.4850E+00
----------------------------
Node 30: Intermediate node
A case goes into Node 60 if zone =
 "3", "5"
```

```
zone mode = "3"
----------------------------
Node 60: Terminal node
Coefficients of loglinear regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -1.9094E+00      -93.79 0.0000
bonus     -2.0022E-01      -53.60 0.0000    1.0000E+00  4.0000E+00  7.0000E+00
Node mean for offset variable =     5.1513E+00
----------------------------
Node 61: Terminal node
Coefficients of loglinear regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -2.0908E+00     -131.72 0.0000
bonus     -2.0319E-01      -70.53 0.0000    1.0000E+00  4.0000E+00  7.0000E+00
Node mean for offset variable =     4.7057E+00
----------------------------
Node 31: Terminal node
Coefficients of loglinear regression function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -2.0319E+00      -64.33 0.0000
bonus     -1.6109E-01      -30.64 0.0000    1.0000E+00  4.0377E+00  7.0000E+00
Node mean for offset variable =     3.5950E+00
----------------------------

Observed and fitted values are stored in sweden.fit
LaTeX code for tree is in sweden.tex
```

The tree is shown in Figure 10.

5.7 Censored response data

GUIDE can fit a piecewise-constant, piecewise-simple linear, or piecewise multiple linear proportional hazards regression model to censored response data.Using usual notation, let $\lambda(\mathbf{x}, t)$ denote the hazard rate at time t for a subject with covariate vector $\mathbf{x}$. In a proportional hazards model, the hazard rate can be factored as $\lambda(\mathbf{x}, t) = \lambda_0(t) f(\mathbf{x}, \boldsymbol{\beta})$, where $\lambda_0(t)$ is a "baseline" hazard rate that is independent of the covariates and $f(\mathbf{x}, \boldsymbol{\beta})$ is a function of $\mathbf{x}$ and some coefficients $\boldsymbol{\beta}$, independent of t. The Cox proportional hazards model uses $\lambda(\mathbf{x}, t) = \lambda_0(t) \exp(\boldsymbol{\beta}'\mathbf{x})$. GUIDE fits the more general model

$$\lambda(\mathbf{x}, t) = \lambda_0(t) \sum_i I(\mathbf{x} \in S_i) \exp(\boldsymbol{\beta}_i' \mathbf{x}),$$

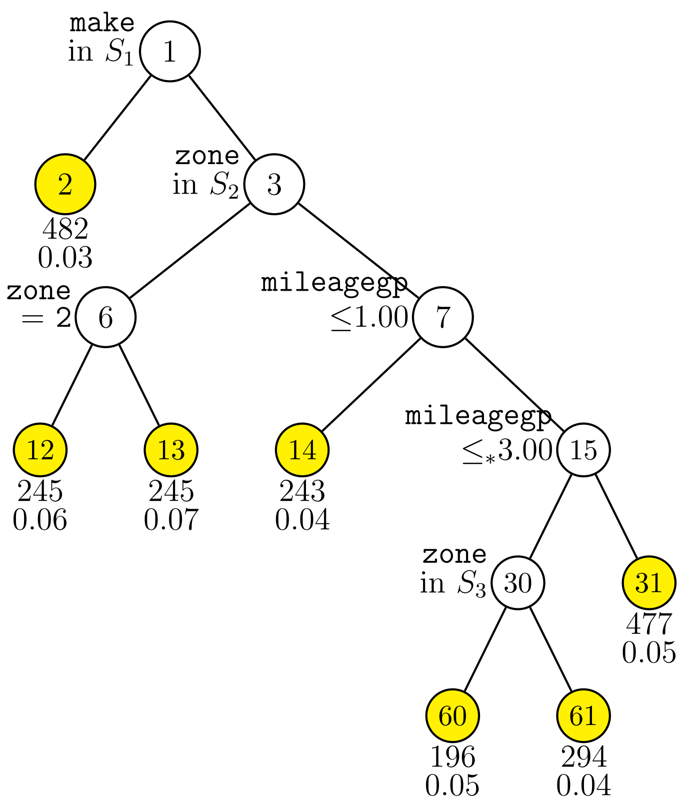

Figure 10: GUIDE 0.50-SE Poisson regression tree for predicting `claims`. At each split, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Set $S_1 = \{4, 6\}$. Set $S_2 = \{1, 2\}$. Set $S_3 = \{3, 5\}$. Sample sizes, sample rates, and names of regressor are printed below nodes.

where S_i is a set corresponding node i and $\boldsymbol{\beta}_i$ is its associated coefficient vector. See Loh et al. (2015) for more details.

We illustrate the piecewise-constant model $\lambda(\mathbf{x}, t) = \lambda_0(t) \sum_i I(\mathbf{x} \in S_i) \exp(\beta_{i0})$ with a data set from the Worcester Heart Attack Study analyzed in Hosmer et al. (2008). The data are in the file `whas500.csv` and the description file in `whas500.dsc` whose contents are repeated below.

```
whas500.csv
NA
1
1 id x
2 age n
3 gender c
4 hr n
5 sysbp n
6 diasbp n
7 bmi n
8 cvd c
9 afb c
10 sho c
11 chf c
12 av3 c
13 miord c
14 mitype c
15 year c
16 admitdate x
17 disdate x
18 fdate x
19 los n
20 dstat x
21 lenfol t
22 fstat d
```

The goal of the study is to observe survival rates following hospital admission for acute myocardial infarction. The response variable is `lenfol`, which stands for total length of follow-up in days. Variable `fstat` is status at last follow-up (0=alive, 1=dead) and variable `chf` is congestive heart complications (0=no, 1=yes).

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning
Input your choice: 1
Name of batch input file: whas500.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
```

```
Name of batch output file: whas500.out
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
Choose type of regression model:
 1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
 5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
Input choice ([1:6], <cr>=1): 4
Choose complexity of model to use at each node:
1: multiple linear, 2: best simple linear, 3: constant ([1:3], <cr>=1): 3
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: whas500.dsc
Reading data description file ...
Training sample file: whas500.csv
Missing value code: NA
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is fstat
Reading data file ...
Number of records in data file: 500
Length of longest data entry: 10
Checking for missing values ...
Total number of cases: 500
Col. no. Categorical variable    #levels    #missing values
       3 gender                        2                  0
       8 cvd                           2                  0
       9 afb                           2                  0
      10 sho                           2                  0
      11 chf                           2                  0
      12 av3                           2                  0
      13 miord                         2                  0
      14 mitype                        2                  0
      15 year                          3                  0
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Smallest uncensored T:  1.0000
No. complete cases excluding censored T < smallest uncensored T: 500
No. cases used to compute baseline hazard: 500
No. cases with D=1 and T >= smallest uncensored: 215
Rereading data
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals    #X-var    #N-var    #F-var    #S-var    #B-var    #C-var
       500          0          0         5         0         0         6         0         9
Survival time variable in column: 21
```

```
Censoring indicator variable in column: 22
Proportion of uncensored among nonmissing T and D variables =  0.430
No. cases used for training: 500
Finish reading data file
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): whas500.tex
A file by that name already exists
Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
Input name of file to store node ID and fitted value of each case: whas500.fit
Input file is created!
Run GUIDE with the command: guide < whas500.in
```

Results

```
Proportional hazards regression with relative risk estimates
Pruning by cross-validation
Data description file: whas500.dsc
Training sample file: whas500.csv
Missing value code: NA
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is fstat
Piecewise constant model
Number of records in data file: 500
Length of longest data entry: 10
Smallest uncensored T:  1.0000
No. complete cases excluding censored T < smallest uncensored T: 500
No. cases used to compute baseline hazard: 500
No. cases with D=1 and T >= smallest uncensored: 215

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
t=survival time variable
 Column  Name            Minimum      Maximum   #Categories       #Missing
      2  age        s   3.0000E+01   1.0400E+02
      3  gender     c                                     2
      4  hr         s   3.5000E+01   1.8600E+02
      5  sysbp      s   5.7000E+01   2.4400E+02
      6  diasbp     s   6.0000E+00   1.9800E+02
      7  bmi        s   1.3045E+01   4.4839E+01
      8  cvd        c                                     2
      9  afb        c                                     2
     10  sho        c                                     2
     11  chf        c                                     2
```

```
     12  av3       c                                    2
     13  miord     c                                    2
     14  mitype    c                                    2
     15  year      c                                    3
     19  los       s   0.0000E+00   4.7000E+01
     21  lenfol    t   1.0000E+00   2.3580E+03
     22  fstat     d   0.0000E+00   1.0000E+00
  ================== Constructed variables ===================
     23  lnbasehaz  z  -4.1352E+00   9.7549E-01

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       500         0          0        5        0        0        6        0        9
 Survival time variable in column: 21
 Censoring indicator variable in column: 22
 Proportion of uncensored among nonmissing T and D variables: 0.430
 No. cases used for training: 500

 Interaction tests on all variables
 Pruning by v-fold cross-validation, with v =  10
 Selected tree is based on mean of CV estimates
 Fraction of cases used for splitting each node =   1.0000
 Max number of split levels =  10
 Minimum node size =  3
 Number of iterations =  5
 Number of SE's for pruned tree =   5.0000E-01

 Size and CV Loss and SE of subtrees:
  Tree   #Tnodes  Mean Loss   SE(Mean)   BSE(Mean)  Median Loss  BSE(Median)
    1       60   1.579E+00   1.039E-01   7.117E-02   1.630E+00   8.633E-02
    2       59   1.579E+00   1.039E-01   7.132E-02   1.632E+00   8.726E-02
    3       58   1.574E+00   1.037E-01   7.088E-02   1.629E+00   8.560E-02
    4       56   1.575E+00   1.037E-01   7.134E-02   1.629E+00   8.698E-02
    5       55   1.576E+00   1.036E-01   7.085E-02   1.623E+00   8.370E-02
    6       54   1.565E+00   1.021E-01   7.149E-02   1.609E+00   8.418E-02
    7       53   1.570E+00   1.023E-01   6.965E-02   1.609E+00   8.278E-02
    8       52   1.573E+00   1.024E-01   6.999E-02   1.609E+00   8.370E-02
    9       49   1.578E+00   1.023E-01   7.098E-02   1.609E+00   8.634E-02
   10       48   1.574E+00   1.023E-01   7.246E-02   1.609E+00   8.744E-02
   11       46   1.574E+00   1.023E-01   7.246E-02   1.609E+00   8.744E-02
   12       45   1.569E+00   1.021E-01   7.195E-02   1.609E+00   8.400E-02
   13       44   1.569E+00   1.018E-01   6.850E-02   1.609E+00   8.305E-02
   14       43   1.561E+00   1.014E-01   6.161E-02   1.604E+00   8.005E-02
   15       42   1.560E+00   1.015E-01   6.124E-02   1.604E+00   7.960E-02
   16       38   1.551E+00   1.013E-01   6.017E-02   1.593E+00   6.745E-02
   17       37   1.556E+00   1.015E-01   6.525E-02   1.558E+00   8.787E-02
```

```
  18       33   1.554E+00   1.014E-01   6.493E-02   1.558E+00   8.577E-02
  19       30   1.553E+00   1.014E-01   6.458E-02   1.558E+00   8.543E-02
  20       29   1.520E+00   9.861E-02   7.153E-02   1.558E+00   8.354E-02
  21       28   1.508E+00   9.803E-02   7.192E-02   1.519E+00   6.852E-02
  22       26   1.504E+00   9.723E-02   7.143E-02   1.516E+00   6.417E-02
  23       25   1.501E+00   9.733E-02   7.324E-02   1.516E+00   6.428E-02
  24       23   1.483E+00   9.532E-02   7.663E-02   1.465E+00   6.276E-02
  25       21   1.478E+00   9.519E-02   7.655E-02   1.454E+00   6.210E-02
  26       20   1.472E+00   9.488E-02   7.686E-02   1.452E+00   6.764E-02
  27       19   1.437E+00   9.287E-02   8.809E-02   1.452E+00   8.111E-02
  28       18   1.389E+00   8.985E-02   8.560E-02   1.376E+00   7.902E-02
  29       12   1.382E+00   8.916E-02   8.459E-02   1.376E+00   7.450E-02
  30       11   1.378E+00   9.014E-02   8.603E-02   1.397E+00   8.716E-02
  31        9   1.329E+00   8.411E-02   6.501E-02   1.422E+00   7.183E-02
  32        7   1.265E+00   7.544E-02   4.533E-02   1.309E+00   6.645E-02
  33*       6   1.238E+00   7.306E-02   3.864E-02   1.257E+00   6.255E-02
  34**      5   1.243E+00   7.171E-02   3.991E-02   1.263E+00   5.840E-02
  35++      4   1.281E+00   7.154E-02   4.064E-02   1.270E+00   4.950E-02
  36        3   1.300E+00   7.015E-02   3.646E-02   1.319E+00   3.915E-02
  37        2   1.325E+00   6.506E-02   2.839E-02   1.313E+00   3.556E-02
  38        1   1.503E+00   5.698E-02   2.544E-02   1.484E+00   3.699E-02

 0-SE tree based on mean is marked with *
 0-SE tree based on median is marked with +
 Selected-SE tree based on mean using naive SE is marked with **
 Selected-SE tree based on mean using bootstrap SE is marked with --
 Selected-SE tree based on median and bootstrap SE is marked with ++
 * tree same as + tree
 ** tree same as -- tree


 Following tree is based on mean CV with naive SE estimate (**).

 Structure of final tree. Each terminal node is marked with a T.

 Rel. risk is mean risk relative to sample average ignoring covariates

 Cases fit give the number of cases used to fit node
 Deviance is mean residual deviance for all cases in node
        Node    Total    Cases Matrix    Node       Node    Split          Other
        label   cases      fit  rank   rel.risk   deviance  variable       variables
           1      500      500     1  1.000E+00  1.505E+00  age
           2      244      244     1  3.726E-01  9.913E-01  chf
           4T      49       49     1  1.110E+00  1.413E+00  miord
           5T     195      195     1  2.124E-01  7.383E-01  year
           3      256      256     1  1.890E+00  1.526E+00  chf
```

```
          6T     106      106     1  3.028E+00  1.372E+00  sho
          7      150      150     1  1.365E+00  1.469E+00  age
         14T     120      120     1  1.063E+00  1.360E+00  los
         15T      30       30     1  3.322E+00  1.278E+00  year

 Number of terminal nodes of final tree: 5
 Total number of nodes of final tree:    9

 Regression tree:

  Node 1: age <=   71.00000
    Node 2: chf = "1"
      Node 4: Risk relative to sample average ignoring covariates =   1.10956
    Node 2: chf /= "1"
      Node 5: Risk relative to sample average ignoring covariates =   0.21235
  Node 1: age >   71.00000 or NA
    Node 3: chf = "1"
      Node 6: Risk relative to sample average ignoring covariates =   3.02760
    Node 3: chf /= "1"
      Node 7: age <=   85.00000 or NA
        Node 14: Risk relative to sample average ignoring covariates =   1.06334
      Node 7: age >   85.00000 and not NA
        Node 15: Risk relative to sample average ignoring covariates =   3.32215

 ***************************************************************


 Node 1: Intermediate node
 A case goes into Node 2 if age <=  7.1000000E+01
 age mean =  6.9846E+01
 Coefficients of log-relative risk function:
 Regressor Coefficient      t-stat  p-val
 Constant  -3.5381E-02       -0.52 0.6041
 Predicted relative risk =   1.0000000000000000
 ----------------------------
 Node 2: Intermediate node
 A case goes into Node 4 if chf = "1"
 chf mode = "0"
 ----------------------------
 Node 4: Terminal node
 Coefficients of log-relative risk function:
 Regressor Coefficient      t-stat  p-val
 Constant   6.8580E-02        0.34 0.7332
 Predicted relative risk =   1.1095574995429367
 ----------------------------
 Node 5: Terminal node
```

```
Coefficients of log-relative risk function:
Regressor Coefficient      t-stat  p-val
Constant  -1.5849E+00       -7.43 0.0000
Predicted relative risk =  0.21235168812700622
----------------------------
Node 3: Intermediate node
A case goes into Node 6 if chf = "1"
chf mode = "0"
----------------------------
Node 6: Terminal node
Coefficients of log-relative risk function:
Regressor Coefficient      t-stat  p-val
Constant   1.0724E+00        9.89 0.0000
Predicted relative risk =   3.0276015801608627
----------------------------
Node 7: Intermediate node
A case goes into Node 14 if age <=  8.5000000E+01 or NA
age mean =  8.0667E+01
----------------------------
Node 14: Terminal node
Coefficients of log-relative risk function:
Regressor Coefficient      t-stat  p-val
Constant   2.6029E-02        0.19 0.8459
Predicted relative risk =   1.0633351387096228
----------------------------
Node 15: Terminal node
Coefficients of log-relative risk function:
Regressor Coefficient      t-stat  p-val
Constant   1.1652E+00        6.05 0.0000
Predicted relative risk =   3.3221527399879980
----------------------------

Observed and fitted values are stored in whas500.fit
LaTeX code for tree is in whas500.tex
```

The tree model, given in Figure 11, shows that risk of death is lowest (0.21 relative to the sample average for the whole data set) for those younger than 72 with no congestive heart complications. The groups with the highest risks (3.0–3.32 relative to average) are those older than 71 with congestive heart complications and those older than 85 without congestive heart complications.

The top 8 lines of the file `whas500.fit` and its column definitions are:

```
train node  survivaltime  logbasecumhaz  relativerisk  survivalprob mediansurvtime
    y   14  2.178000E+03  -7.667985E-02  1.063335E+00  3.865048E-01   1.553841E+03
    y    5  2.172000E+03  -7.667985E-02  2.123517E-01  8.270912E-01   2.354932E+03
```

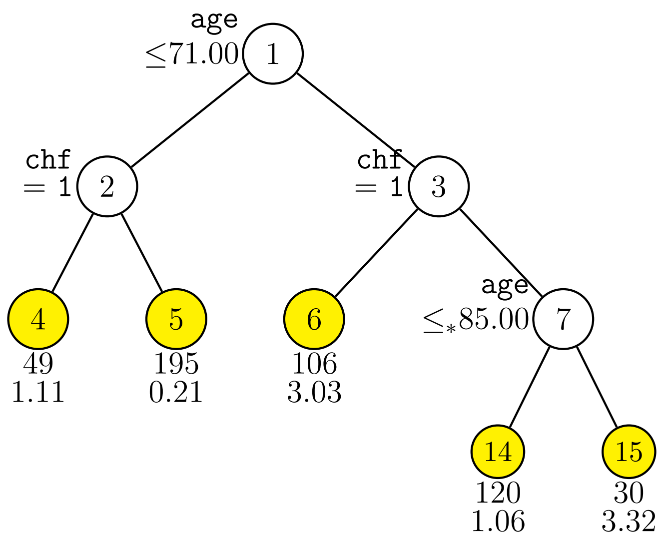

Figure 11: GUIDE 0.50-SE piecewise constant relative risk regression tree for predicting `fstat`. At each split, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Sample sizes and mean relative risks (relative to sample average ignoring covariates) are printed below nodes.

```
    y    5  2.190000E+03  -7.667985E-02  2.123517E-01  8.270912E-01   2.354932E+03
    y    4  2.970000E+02  -1.320296E+00  1.109557E+00  7.512523E-01   1.534972E+03
    y    5  2.131000E+03  -2.213734E-01  2.123517E-01  8.485159E-01   2.354932E+03
    y    5  1.000000E+00  -4.352824E+00  2.123517E-01  9.973654E-01   2.354932E+03
    y    5  2.122000E+03  -2.213734E-01  2.123517E-01  8.485159E-01   2.354932E+03
    y    5  1.496000E+03  -4.919833E-01  2.123517E-01  8.822135E-01   2.354932E+03
```

train: "y" if the observation is used for model fitting, "n" if not.

node: terminal node label of observation.

survivaltime: observed survival time t.

logbasecumhaz: log of the estimated baseline cumulative hazard function $\log \Lambda_0(t) = \log \int_0^t \lambda_0(u)\, du$ at observed time t.

relativerisk: $\exp(\boldsymbol{\beta}'\mathbf{x} - \beta_*)$, risk of death relative to the average for the sample, where $\mathbf{x}$ is the covariate vector of the observation, $\boldsymbol{\beta}$ is the estimated regression coefficient vector in the node, and β_* is the coefficient in the constant model $\lambda_0(t)\exp(\beta_*)$ fitted to all the training cases in the root node. Because a constant is fitted to each node here, $\beta_* = -0.035381$ is the value of $\boldsymbol{\beta}$ at the root node. For example, the first subject, which is in node 14, has $\boldsymbol{\beta} = 0.026029$ and so `relativerisk` $= \exp(\boldsymbol{\beta} - \beta_*) = \exp(0.026029 + 0.035381) = 1.063335$.

`survivalprob`: probability that the subject survives up to observed time t. For the first subject, this is

$$\begin{aligned}\exp\{-\Lambda_0(t)\exp(\boldsymbol{\beta}'\mathbf{x})\} &= \exp\{-\exp(\beta_* + \texttt{logbasecumhaz}) \times \texttt{relativerisk}\} \\ &= \exp(-\exp(-0.035381 - 0.07667985) \times 1.063335) \\ &= 0.3865048.\end{aligned}$$

`mediansurvtime`: estimated median survival time t such that $\exp\{-\Lambda_0(t)\exp(\boldsymbol{\beta}'\mathbf{x})\} = 0.5$, or, equivalently, $\Lambda_0(t)\exp(\boldsymbol{\beta}_i'\mathbf{x}) = -\log(0.5)$, or `logbasecumhaz`$(t) = \log\log(2) - \boldsymbol{\beta}_i'\mathbf{x}$, using linear interpolation of $\Lambda_0(t)$. Median survival times greater than the largest observed time have a trailing plus (+) sign. Figure 12 shows plots of $\log\Lambda_0(t)$ and $\Lambda_0(t)$ for this data set.

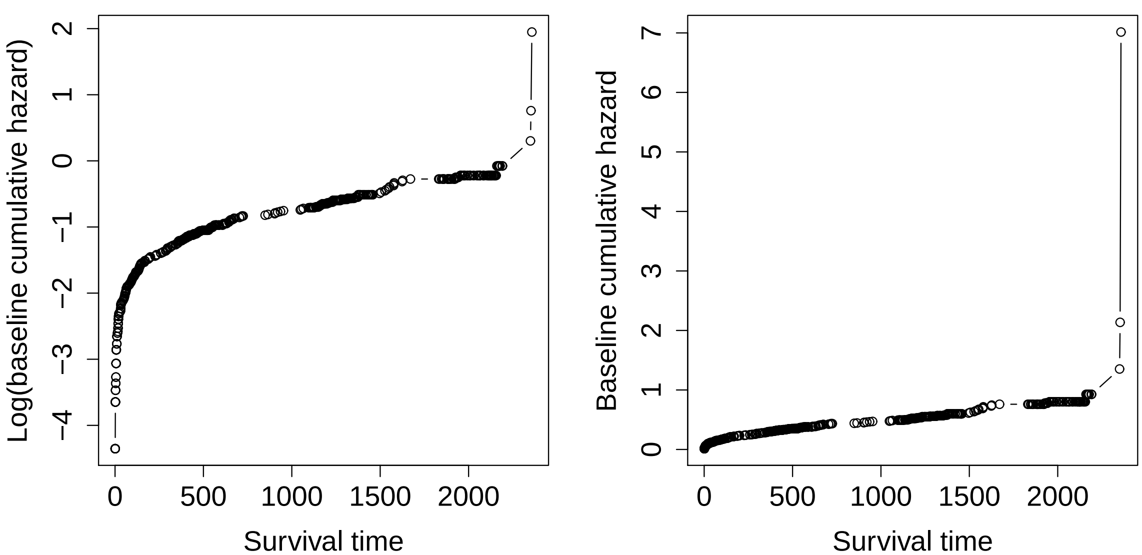

Figure 12: Plots of estimated baseline cumulative hazard function

5.8 Multi-response data

GUIDE can fit a piecewise-constant regression model for two or more dependent variables simultaneously. Following is an example from Loh and Zheng (2013) on estimating the strength and viscosity of concrete. The comma-delimited data file `concrete.csv` is from Yeh (2007). The data description file `concretedsc.txt` is below. Notice that there are three `D` variables. Our goal is to construct a single regression tree that predicts all three `D` variables simultaneously.

```
concrete.csv
NA
1
1 No x
2 Cement n
3 Slag n
4 FlyAsh n
5 Water n
6 SP n
7 CoarseAggr n
8 FineAggr n
9 Slump d
10 Flow d
11 Strength d
```

Following is an annotated log of the input file creation.

```
 0. Read the warranty disclaimer
 1. Create an input file for batch run
 2. Fit a model without creating input file
 3. Convert data to other formats
 4. Variable importance scoring and differential item functioning without input file creation
 Input your choice: 1
 Name of batch input file: concrete.in
 File concrete.in exists
 Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
 Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
 Name of batch output file: concrete.out
 File concrete.out exists
 Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
 Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
 Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
 Choose type of regression model:
  1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
  5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
 Input choice ([1:6], <cr>=1): 5
  Option 5 is for multiresponse data.
 Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

 Input name of data description file (max 100 characters);
 enclose with matching quotes if it has spaces: concretedsc.txt
 Reading data description file ...
```

```
Training sample file: concrete.csv
Missing value code: NA
Records in data file start on line 1
Warning: N variables changed to S
Number of D variables = 3
D variables are:
Slump
Flow
Strength
Choose multivariate or univariate split variable selection:
Choose multivariate if there is an order among the D variables; otherwise choose univariate
Input 1 for multivariate, 2 for univariate ([1:2], <cr>=1): 2
Input 1 to normalize D variables, 2 for no normalization ([1:2], <cr>=1):
Normalization scales each variable to have variance 1.
Input 1 for equal, 2 for unequal weighting of D variables ([1:2], <cr>=1):
Reading data file ...
Number of records in data file: 103
Length of longest data entry: 6
Checking for missing values ...
Total number of cases: 103
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Normalizing data
Rereading data
PCA can be used for variable selection
Do not use PCA if differential item functioning (DIF) scores are wanted
Input 1 to use PCA, 2 otherwise ([1:2], <cr>=2):
#cases w/ miss. D = number of cases with all D values missing
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       103          0          0        1        0        0        7        0        0
No. cases used for training: 103
Finish reading data file
Warning: interaction tests skipped
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): concrete.tex
Input 2 to save node IDs of individual cases, 1 otherwise ([1:2], <cr>=2):
Input name of file to store terminal node ID of each case: concrete.nid
Input 2 to save fitted values at each terminal node; 1 otherwise ([1:2], <cr>=1):
Input file is created!
Run GUIDE with the command: guide < concrete.in
```

Results

```
Multi-response or longitudinal data without T variables
Pruning by cross-validation
Data description file: concretedsc.txt
Training sample file: concrete.csv
Missing value code: NA
Records in data file start on line 1
Warning: N variables changed to S
Number of D variables = 3
Univariate split variable selection method
Mean-squared errors (MSE) are calculated from normalized D variables
 This is a reminder that the D variables are normalized.
D variables equally weighted
Piecewise constant model
Number of records in data file: 103
Length of longest data entry: 6
Neither LDA nor PCA used

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
 Column  Name             Minimum      Maximum   #Categories       #Missing
      2  Cement      s   1.3700E+02   3.7400E+02
      3  Slag        s   0.0000E+00   1.9300E+02
      4  FlyAsh      s   0.0000E+00   2.6000E+02
      5  Water       s   1.6000E+02   2.4000E+02
      6  SP          s   4.4000E+00   1.9000E+01
      7  CoarseAggr  s   7.0800E+02   1.0499E+03
      8  FineAggr    s   6.4060E+02   9.0200E+02
      9  Slump       d   0.0000E+00   2.9000E+01
     10  Flow        d   2.0000E+01   7.8000E+01
     11  Strength    d   1.7190E+01   5.8530E+01

#cases w/ miss. D = number of cases with all D values missing
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       103          0          0        1        0        0        7        0        0
No. cases used for training: 103

Warning: interaction tests skipped
No interaction tests
Pruning by v-fold cross-validation, with v =  10
Selected tree is based on mean of CV estimates
Split values for N and S variables based on exhaustive search
Max number of split levels =  10
Minimum node size =  10
```

```
Pruning sequence
  Subtree     Pruned  #Terminal        True           Geometric
   number       node      nodes        alpha             mean
        0          0          7       0.0000          0.0000
        1         14          6      0.13747E-01     0.15854E-01
        2         15          5      0.18284E-01     0.37205E-01
        3          3          3      0.75707E-01     0.85456E-01
        4          2          2      0.96461E-01     0.15729
        5          1          1      0.25649         0.17977+309

Number of SE's for pruned tree =   5.0000E-01

Size and CV Loss and SE of subtrees:
 Tree   #Tnodes  Mean Loss   SE(Mean)   BSE(Mean)  Median Loss  BSE(Median)
   0        7   7.119E-01   8.938E-02   9.007E-02   6.070E-01   1.436E-01
   1+       6   7.144E-01   8.884E-02   8.968E-02   6.070E-01   1.413E-01
   2**      5   7.039E-01   8.780E-02   8.923E-02   6.145E-01   1.240E-01
   3        3   8.090E-01   9.190E-02   9.209E-02   6.778E-01   1.114E-01
   4        2   7.983E-01   8.096E-02   6.855E-02   7.304E-01   7.444E-02
   5        1   1.011E+00   9.866E-02   7.561E-02   9.799E-01   1.073E-01

0-SE tree based on mean is marked with *
0-SE tree based on median is marked with +
Selected-SE tree based on mean using naive SE is marked with **
Selected-SE tree based on mean using bootstrap SE is marked with --
Selected-SE tree based on median and bootstrap SE is marked with ++
** tree same as ++ tree
** tree same as -- tree
++ tree same as -- tree
* tree same as ** tree
* tree same as ++ tree
* tree same as -- tree


Following tree is based on mean CV with naive SE estimate (**).

Structure of final tree. Each terminal node is marked with a T.

Cases fit give the number of cases used to fit node
MSE is residual sum of squares divided by number of cases in node
       Node    Total    Cases      Node    Split
      label    cases      fit       MSE    variable
          1      103      103  1.000E+00   Water
          2       29       29  9.453E-01   CoarseAggr
          4T      17       17  7.837E-01   -
          5T      12       12  3.632E-01   -
```

```
        3       74       74  6.728E-01   FlyAsh
       6T       19       19  6.053E-01   -
        7       55       55  5.866E-01   Cement
      14T       24       24  4.157E-01   Cement
      15T       31       31  4.354E-01   Cement

Number of terminal nodes of final tree: 5
Total number of nodes of final tree:    9

Regression tree for multi-response data:

 Node 1: Water <=  1.82250E+02
   Node 2: CoarseAggr <=  9.60500E+02 or NA
     Node 4: Mean cost =  7.37564E-01
   Node 2: CoarseAggr >  9.60500E+02 and not NA
     Node 5: Mean cost =  3.32963E-01
 Node 1: Water >  1.82250E+02 or NA
   Node 3: FlyAsh <=   48.50000
     Node 6: Mean cost =  5.73405E-01
   Node 3: FlyAsh >   48.50000 or NA
     Node 7: Cement <=  1.65650E+02
       Node 14: Mean cost =  3.98351E-01
     Node 7: Cement >  1.65650E+02 or NA
       Node 15: Mean cost =  4.21329E-01

***************************************************************


Node 1: Intermediate node
A case goes into Node 2 if Water <=  1.8225000E+02
Water mean =  1.9717E+02
Estimated D values are:
  1.8049E+01   4.9611E+01   3.6039E+01
```

The three numbers are the sample mean values of Slump, Flow and Strength, in order of their appearance in the description file.

```
----------------------------
Node 2: Intermediate node
A case goes into Node 4 if CoarseAggr <=  9.6050000E+02 or NA
CoarseAggr mean =  9.5512E+02
----------------------------
Node 4: Terminal node
Estimated D values are:
  5.5294E+00   2.6747E+01   4.4365E+01
----------------------------
Node 5: Terminal node
Estimated D values are:
```

```
  1.7083E+01   4.1083E+01   3.3768E+01
 ----------------------------
 Node 3: Intermediate node
 A case goes into Node 6 if FlyAsh <=  4.8500000E+01
 FlyAsh mean =  1.2926E+02
 ----------------------------
 Node 6: Terminal node
 Estimated D values are:
   1.8289E+01   4.9368E+01   2.8884E+01
 ----------------------------
 Node 7: Intermediate node
 A case goes into Node 14 if Cement <=  1.6565000E+02
 Cement mean =  2.2766E+02
 ----------------------------
 Node 14: Terminal node
 Estimated D values are:
   2.1021E+01   5.6083E+01   3.0345E+01
 ----------------------------
 Node 15: Terminal node
 Estimated D values are:
   2.2839E+01   6.0587E+01   4.1147E+01
 ----------------------------

 Case and node IDs are in file: concrete.nid
 Observed and fitted values are stored in concrete.nid
 LaTeX code for tree is in concrete.tex
```

The LaTeX tree is shown in Figure 13.

5.9 Longitudinal data with irregular time points: non-default options

The data come from a longitudinal study on the hourly wage of 888 male high-school dropouts (246 black, 204 Hispanic, 438 white), where the observation time points as well as their number (1–13) varied across individuals (Murnane et al., 1999; Singer and Willett, 2003). An earlier version of GUIDE was used to analyze these data in Loh and Zheng (2013).

The response variable is hourly wage (in 1990 dollars) and the predictor variables are `hgc` (highest grade completed; 6–12), `exper` (years in labor force; 0.001–12.7 yrs), and `race` (Black, Hispanic, and White). The data file `wagedat.txt` is in *wide format*, where each record refers to one individual. The description file `wagedsc.txt` is given below. Note that observation time points are marked as `t`.

`wagedat.txt`

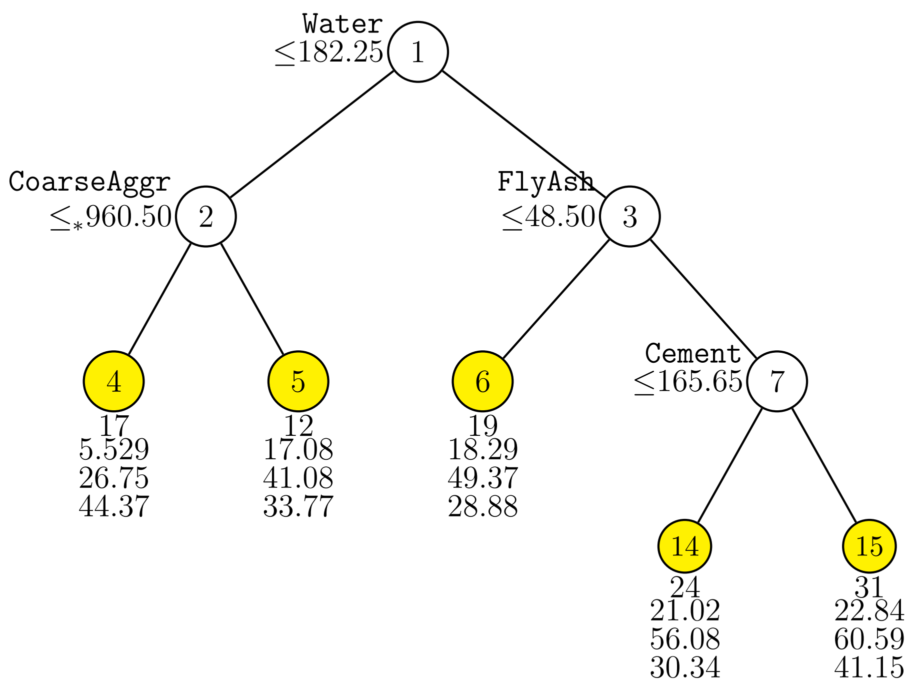

Figure 13: GUIDE 0.50-SE regression tree for predicting response variables `Slump`, `Flow`, and `Strength`. At each split, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Sample sizes and predicted values of `Slump`, `Flow`,and `Strength` are printed below nodes.

```
NA
1
1 id x
2 hgc n
3 exper1 t
4 exper2 t
5 exper3 t
6 exper4 t
7 exper5 t
8 exper6 t
9 exper7 t
10 exper8 t
11 exper9 t
12 exper10 t
13 exper11 t
14 exper12 t
15 exper13 t
16 postexp1 x
17 postexp2 x
18 postexp3 x
19 postexp4 x
20 postexp5 x
21 postexp6 x
22 postexp7 x
23 postexp8 x
24 postexp9 x
25 postexp10 x
26 postexp11 x
27 postexp12 x
28 postexp13 x
29 wage1 d
30 wage2 d
31 wage3 d
32 wage4 d
33 wage5 d
34 wage6 d
35 wage7 d
36 wage8 d
37 wage9 d
38 wage10 d
39 wage11 d
40 wage12 d
41 wage13 d
42 ged1 x
43 ged2 x
44 ged3 x
```

```
45 ged4 x
46 ged5 x
47 ged6 x
48 ged7 x
49 ged8 x
50 ged9 x
51 ged10 x
52 ged11 x
53 ged12 x
54 ged13 x
55 uerate1 x
56 uerate2 x
57 uerate3 x
58 uerate4 x
59 uerate5 x
60 uerate6 x
61 uerate7 x
62 uerate8 x
63 uerate9 x
64 uerate10 x
65 uerate11 x
66 uerate12 x
67 uerate13 x
68 race c
```

Because the default 0.5-SE rule yields a trivial tree with no splits, we show how the options can be changed to produce a tree with the 0-SE rule. Following is a session log.

```
 0. Read the warranty disclaimer
 1. Create an input file for batch run
 2. Fit a model without creating input file
 3. Convert data to other formats
 4. Variable importance scoring and differential item functioning
 Input your choice: 1
 Name of batch input file: wage.in
 Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
 Name of batch output file: wage.out
 Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
 Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
 Choose type of regression model:
  1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
  5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
 Input choice ([1:6], <cr>=1): 6
 Input 1 for lowess smoothing, 2 for spline smoothing ([1:2], <cr>=1):
 Input 1 for default options, 2 otherwise ([1:2], <cr>=1): 2
```

```
 Choosing 1 will produce a 0.5-SE tree. We choose 2 to allow more options.
Input 1 for interaction tests, 2 to skip them ([1:2], <cr>=1):
Input 1 to prune by CV, 2 for no pruning ([1:2], <cr>=1):

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: wagedsc.txt
Reading data description file ...
Training sample file: wagedat.txt
Missing value code: NA
Records in data file start on line 1
Warning: N variables changed to S
Number of D variables = 13
Number of D variables =          13
D variables are:
wage1
wage2
wage3
wage4
wage5
wage6
wage7
wage8
wage9
wage10
wage11
wage12
wage13
T variables are:
exper1
exper2
exper3
exper4
exper5
exper6
exper7
exper8
exper9
exper10
exper11
exper12
exper13
The D variables can be grouped into segments to look for patterns
Input 1 for roughly equal-sized groups, 2 for customized groups ([1:2], <cr>=1):
Input number of roughly equal-sized groups ([2:9], <cr>=3):
Input number of interpolating points for prediction ([10:100], <cr>=31):
Reading data file ...
```

```
Number of records in data file: 888
Length of longest data entry: 16
Checking for missing values ...
Total number of cases: 888
Col. no. Categorical variable    #levels    #missing values
      68 race                          3                  0
Re-checking data ...
Allocating missing value information
Assigning codes to categorical and missing values
Finish checking data
Creating missing value indicators
Rereading data
#cases w/ miss. D = number of cases with all D values missing
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       888          0          0       40        0        0        1        0        1
No. cases used for training: 888
No. cases excluded due to 0 weight or missing D: 0
Finish reading data file
Warning: interaction tests skipped
Default number of cross-validations =           10
Input 1 to accept the default, 2 to change it ([1:2], <cr>=1):
Best tree may be chosen based on mean or median CV estimate
Input 1 for mean-based, 2 for median-based ([1:2], <cr>=1):
Input number of SEs for pruning ([0.00:1000.00], <cr>=0.50): 0
 This is where we choose the 0-SE pruning rule.
Choose a split point selection method for numerical variables:
Choose 1 to use faster method based on sample quantiles
Choose 2 to use exhaustive search
Input 1 or 2 ([1:2], <cr>=2):
Default max number of split levels =           10
Input 1 to accept this value, 2 to change it ([1:2], <cr>=1):
Default minimum node sample size is 44
Input 1 to use the default value, 2 to change it ([1:2], <cr>=1):
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): wage.tex
A file by that name already exists
Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
Input 1 for a vertical tree, 2 for a sideways tree ([1:2], <cr>=1):
Input 1 to include node numbers, 2 to omit them ([1:2], <cr>=1):
Input 1 to number all nodes, 2 to number leaves only ([1:2], <cr>=1):
Choose a color for the terminal nodes:
(1) white
(2) lightgray
(3) gray
(4) darkgray
```

```
(5) black
(6) yellow
(7) red
(8) blue
(9) green
(10) magenta
(11) cyan
Input your choice ([1:11], <cr>=6):
You can store the variables and/or values used to split and fit in a file
Choose 1 to skip this step, 2 to store split and fit variables,
3 to store split variables and their values
Input your choice ([1:3], <cr>=1):
Input 2 to save node IDs of individual cases, 1 otherwise ([1:2], <cr>=2):
Input name of file to store terminal node ID of each case: wage.nid
A file by that name exists
Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
Input 2 to save fitted values at each terminal node; 1 otherwise ([1:2], <cr>=1):
Input 2 to save terminal node IDs for importance scoring; 1 otherwise ([1:2], <cr>=1):
Input 2 to write R function for predicting new cases, 1 otherwise ([1:2], <cr>=1):
Input file is created!
Run GUIDE with the command: guide < wage.in
```

Results

```
Lowess smoothing
Longitudinal data with T variables
Pruning by cross-validation
Data description file: wagedsc.txt
Training sample file: wagedat.txt
Missing value code: NA
Records in data file start on line 1
Warning: N variables changed to S
Number of D variables = 13
Number of D variables =  13
D variables are:
wage1
wage2
wage3
wage4
wage5
wage6
wage7
wage8
wage9
wage10
```

```
wage11
wage12
wage13
T variables are:
exper1
exper2
exper3
exper4
exper5
exper6
exper7
exper8
exper9
exper10
exper11
exper12
exper13
Number of records in data file: 888
Length of longest data entry: 16

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
 Column  Name            Minimum      Maximum   #Categories       #Missing
      2  hgc        s   6.0000E+00   1.2000E+01
      3  exper1     t   1.0000E-03   5.6370E+00
      4  exper2     t   0.0000E+00   7.5840E+00                         38
      5  exper3     t   0.0000E+00   9.7770E+00                         77
      6  exper4     t   0.0000E+00   1.0815E+01                        124
      7  exper5     t   0.0000E+00   1.1777E+01                        159
      8  exper6     t   0.0000E+00   1.0587E+01                        233
      9  exper7     t   0.0000E+00   1.1279E+01                        325
     10  exper8     t   0.0000E+00   1.0582E+01                        428
     11  exper9     t   0.0000E+00   1.1621E+01                        551
     12  exper10    t   0.0000E+00   1.2260E+01                        678
     13  exper11    t   0.0000E+00   1.1980E+01                        791
     14  exper12    t   0.0000E+00   1.2558E+01                        856
     15  exper13    t   0.0000E+00   1.2700E+01                        882
     29  wage1      d   2.0299E+00   6.8649E+01
     30  wage2      d   2.0689E+00   5.0400E+01                         38
     31  wage3      d   2.0462E+00   3.4501E+01                         77
     32  wage4      d   2.1170E+00   3.3149E+01                        124
     33  wage5      d   2.1043E+00   4.9304E+01                        159
     34  wage6      d   2.2078E+00   7.3995E+01                        233
     35  wage7      d   2.1043E+00   4.7276E+01                        325
     36  wage8      d   2.3164E+00   3.7713E+01                        428
```

```
     37  wage9      d   2.5294E+00   4.6109E+01                          551
     38  wage10     d   2.9982E+00   5.6543E+01                          678
     39  wage11     d   4.0837E+00   2.2198E+01                          791
     40  wage12     d   3.4315E+00   4.6201E+01                          856
     41  wage13     d   4.5631E+00   7.7757E+00                          882
     68  race       c                                        3

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       888         0          0       40        0        0        1        0        1
 No. cases used for training: 888
 No. cases excluded due to 0 weight or missing D: 0

 Warning: interaction tests skipped
 No interaction tests
 Pruning by v-fold cross-validation, with v =  10
 Selected tree is based on mean of CV estimates
 Split values for N and S variables based on exhaustive search
 Max number of split levels =  10
 Minimum node size =  44

 Pruning sequence
   Subtree     Pruned  #Terminal        True           Geometric
    number       node      nodes        alpha             mean
         0          0         10       0.0000          0.0000
         1          4          9       0.0000          0.0000
         2         15          8       0.0000          0.0000
         3         14          6      0.28144E-01     0.65129E-01
         4          2          5      0.15072         0.30744
         5          1          1      0.62712         0.17977+309

 Number of SE's for pruned tree =   0.0000E+00

 Size and CV Loss and SE of subtrees:
  Tree   #Tnodes  Mean Loss   SE(Mean)   BSE(Mean)  Median Loss  BSE(Median)
    0       10   1.257E+02   1.044E+01   8.502E+00   1.204E+02   1.525E+01
    1        9   1.257E+02   1.044E+01   8.502E+00   1.204E+02   1.525E+01
    2        8   1.257E+02   1.044E+01   8.502E+00   1.204E+02   1.525E+01
    3        6   1.242E+02   1.049E+01   8.463E+00   1.181E+02   1.537E+01
    4**      5   1.238E+02   1.058E+01   8.434E+00   1.175E+02   1.530E+01
    5++      1   1.244E+02   1.064E+01   8.700E+00   1.157E+02   1.577E+01

 0-SE tree based on mean is marked with *
 0-SE tree based on median is marked with +
 Selected-SE tree based on mean using naive SE is marked with **
 Selected-SE tree based on mean using bootstrap SE is marked with --
```

```
Selected-SE tree based on median and bootstrap SE is marked with ++
** tree same as -- tree
+ tree same as ++ tree
* tree same as ** tree
* tree same as -- tree


Following tree is based on mean CV with naive SE estimate (**).

Structure of final tree. Each terminal node is marked with a T.

Cases fit give the number of cases used to fit node
MSE is residual sum of squares divided by number of cases in node
       Node    Total    Cases      Node    Split
      label    cases      fit       MSE    variable
          1      888      888  1.222E+02   race
          2T     246      246  1.111E+02   hgc
          3      642      642  1.259E+02   race
          6      204      204  1.278E+02   hgc
         12T     127      127  1.085E+02   -
         13T      77       77  1.514E+02   -
          7      438      438  1.252E+02   hgc
         14T     299      299  9.813E+01   hgc
         15T     139      139  1.777E+02   hgc

Number of terminal nodes of final tree: 5
Total number of nodes of final tree:    9

Regression tree for longitudinal data:

 Node 1: race = "black"
   Node 2: Mean cost =  1.10602E+02
 Node 1: race /= "black"
   Node 3: race = "hispanic"
     Node 6: hgc <=    9.50000 or NA
       Node 12: Mean cost =  1.07621E+02
     Node 6: hgc >    9.50000 and not NA
       Node 13: Mean cost =  1.49412E+02
   Node 3: race /= "hispanic"
     Node 7: hgc <=    9.50000 or NA
       Node 14: Mean cost =  9.78002E+01
     Node 7: hgc >    9.50000 and not NA
       Node 15: Mean cost =  1.76394E+02

***************************************************************
```

```
Node 1: Intermediate node
A case goes into Node 2 if race = "black"
race mode = "white"
----------------------------
Node 2: Terminal node
----------------------------
Node 3: Intermediate node
A case goes into Node 6 if race = "hispanic"
race mode = "white"
----------------------------
Node 6: Intermediate node
A case goes into Node 12 if hgc <=  9.5000000E+00 or NA
hgc mean =  8.9118E+00
----------------------------
Node 12: Terminal node
----------------------------
Node 13: Terminal node
----------------------------
Node 7: Intermediate node
A case goes into Node 14 if hgc <=  9.5000000E+00 or NA
hgc mean =  8.8973E+00
----------------------------
Node 14: Terminal node
----------------------------
Node 15: Terminal node
----------------------------

Case and node IDs are in file: wage.nid
Observed and fitted values are stored in wage.nid
LaTeX code for tree is in wage.tex
```

Figure 14 shows the tree and Figure 15 plots lowess-smoothed curves of mean wage in the two terminal nodes. The plotting values are obtained from the result file `wage.fit` whose contents are given below. The first column gives the node number and the next two columns the start and end of the times at which fitted values are computed. The other columns give the fitted values equally spaced between the start and end times.

```
node t.start t.end fitted1 fitted2 fitted3 fitted4 fitted5 fitted6 fitted7 fitted8 fitted9 fitted10
   2   0.40000E-02  0.12558E+02  0.50794E+01  0.52623E+01  0.54112E+01  0.55477E+01  0.56649E+01  0
  12   0.60000E-02  0.12535E+02  0.47994E+01  0.50688E+01  0.53388E+01  0.55076E+01  0.54340E+01  0
  13   0.12200E+00  0.11990E+02  0.56361E+01  0.58877E+01  0.61037E+01  0.62417E+01  0.62780E+01  0
  14   0.10000E-02  0.12700E+02  0.50837E+01  0.52324E+01  0.53638E+01  0.54764E+01  0.56314E+01  0
```

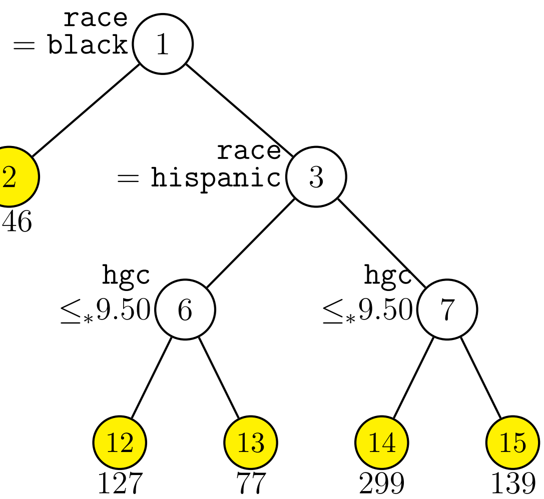

Figure 14: GUIDE 0.00-SE regression tree for predicting longitudinal variables `wage1`, `wage2`, etc. At each split, an observation goes to the left branch if and only if the condition is satisfied. The symbol '$\leq_*$' stands for '$\leq$ or missing'. Sample sizes are printed below nodes.

```
  15   0.20000E-02  0.12045E+02  0.57487E+01  0.59247E+01  0.60944E+01  0.62441E+01  0.64134E+01
```

5.10 Subgroup identification

If there is a treatment variable in the data, GUIDE can fit a tree model find subgroups with differential treatment effects. The dependent variable can be censored or not.

1. The treatment variable is designated as `R`.
2. If there is no censoring, the response variable is designated as `D` as usual.
3. If there is censoring, then the survival time is designated as `T` and the event (typically death) indicator is designated as `D`, taking value 1 for death and 0 for censored.

GUIDE has two methods for solving this problem, called `gi` and `gs`. The former is more sensitive to predictive variables (i.e., variables that interact with treatment) and the latter is equally sensitive to prognostic and predictive variables. The methods are documented in Loh et al. (2015).

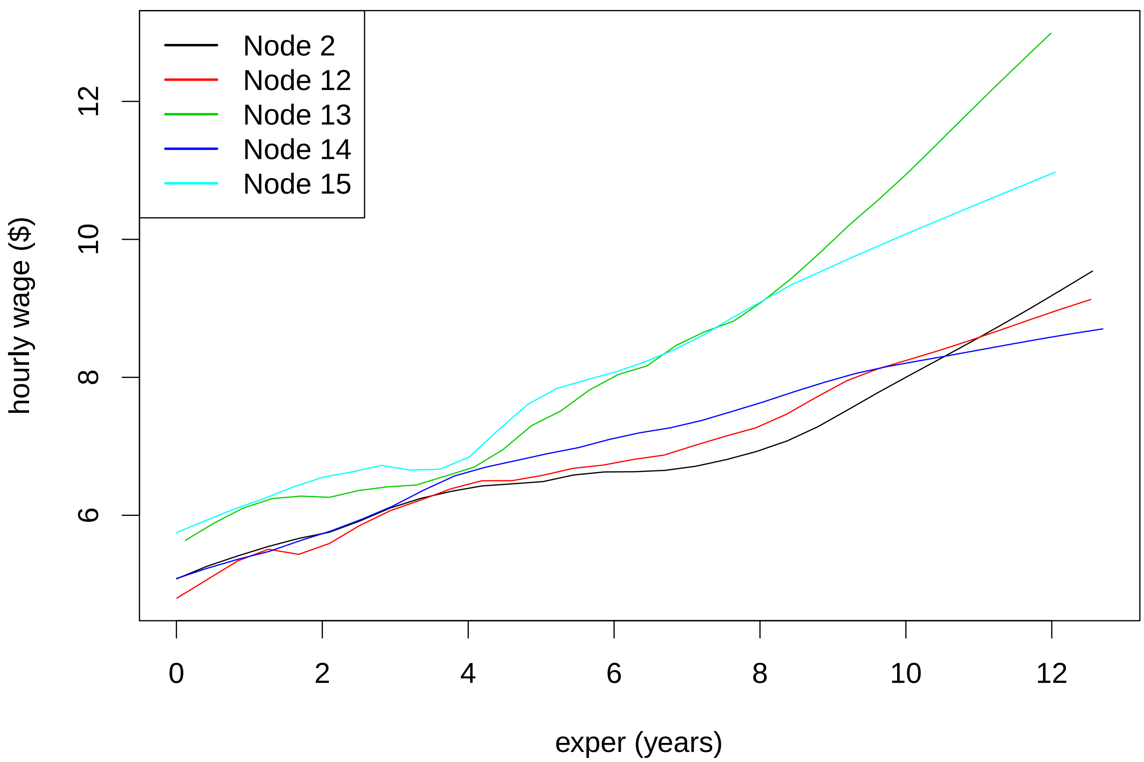

Figure 15: Lowess-smoothed mean wage curves in the terminal nodes of Figure 14.

We illustrate the gi method with data from a breast cancer trial (Schmoor et al., 1996). The data are in the file cancer.txt from the ipred R package (Peters and Hothorn, 2012). In the description file cancerdsc.txt below, the treatment variable is hormone therapy, horTh. The variable time is censored survival time and death is the event indicator (1=death, 0=censored).

```
cancer.txt
NA
1
1 horTh r
2 age s
3 menostat c
4 tsize s
5 tgrade c
6 pnodes s
7 progrec s
8 estrec s
9 time t
10 death d
```

Input file generation

```
 0. Read the warranty disclaimer
 1. Create an input file for batch run
 2. Fit a model without creating input file
 3. Convert data to other formats
 4. Variable importance scoring and differential item functioning
 Input your choice: 1
 Name of batch input file: cancerin.txt
 Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
 Name of batch output file: cancerout.txt
 Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
 Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
 Choose type of regression model:
  1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
  5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
 Input choice ([1:6], <cr>=1): 4
 Choose complexity of model to use at each node:
 1: multiple linear, 2: best simple linear, 3: constant ([1:3], <cr>=1):
 Input 1 for default options, 2 otherwise ([1:2], <cr>=1):
 Input name of data description file (max 100 characters);
 enclose with matching quotes if it has spaces: cancerdsc.txt
 Reading data description file ...
 Training sample file: cancer.txt
 Missing value code: NA
```

```
Records in data file start on line 1
 R variable present
Dependent variable is death
Reading data file ...
Number of records in data file: 686
Length of longest data entry: 4
Checking for missing values ...
Total number of cases: 686
Col. no. Categorical variable    #levels    #missing values
       1 horTh                         2                  0
       3 menostat                      2                  0
       5 tgrade                        3                  0
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Smallest uncensored T: 72.00
No. cases dropped due to missing D or T or censored T < smallest uncensored T =  14
No. complete cases excluding censored T < smallest uncensored T: 672
No. cases used to compute baseline hazard: 672
No. cases with D=1 and T >= smallest uncensored: 299
GUIDE will try to create the variables in the description file.
If it is unsuccessful, please create the columns yourself...
Number of dummy variables created:             1
Choose a subgroup identification method:
1 = Sum of chi-squares (Gs)
2 = Treatment interactions (Gi)
Input your choice: ([1:2], <cr>=2):
 Option 2 is generally more sensitive to detecting treatment interactions.
Creating dummy variables
Rereading data
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals    #X-var    #N-var    #F-var    #S-var    #B-var    #C-var    #R-var
       686          0          0         0         0         0         5         0         2         1
Survival time variable in column: 9
Censoring indicator variable in column: 10
Proportion of uncensored among nonmissing T and D variables =  0.445
No. cases used for training: 672
Finish reading data file
Warning: interaction tests skipped
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): cancer.tex
A file by that name already exists
Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
Input 2 to save individual fitted values and node IDs, 1 otherwise ([1:2], <cr>=2):
Input name of file to store node ID and fitted value of each case: cancer.fit
A file by that name exists
```

```
Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
Input file is created!
Run GUIDE with the command: guide < cancerin.txt

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals    #X-var   #N-var   #F-var   #S-var   #B-var   #C-var   #R-var
       686          0          0         0        0        0        5        0        2        1
Survival time variable in column:           9
Censoring indicator variable in column:          10
Proportion of uncensored among nonmissing T and D variables =   0.445
No. cases used for training =          672
Warning: interaction tests skipped
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): cancer.tex
Input 2 to save fitted values and node IDs, 1 otherwise ([1:2], <cr>=1): 2
Input name of file to store node IDs and fitted values: cancer.fit
Input file is created!
```

Results The following results show that the tree splits once on variable `progrec`.

```
Proportional hazards regression with relative risk estimates
No truncation of predicted values
Pruning by cross-validation
Data description file: cancerdsc.txt
Training sample file: cancer.txt
Missing value code: NA
Records in data file start on line 1
R variable present
Dependent variable is death
Piecewise linear model
Number of records in data file: 686
Length of longest data entry: 4
Smallest uncensored T: 72.00
No. cases dropped due to missing D or T or censored T < smallest uncensored T =  14
No. complete cases excluding censored T < smallest uncensored T: 672
No. cases used to compute baseline hazard: 672
No. cases with D=1 and T >= smallest uncensored: 299
Number of dummy variables created =  1

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
t=survival time variable
 Column  Name            Minimum      Maximum   #Categories       #Missing
      1  horTh      r                                     2
```

```
      2  age        s   2.1000E+01   8.0000E+01
      3  menostat   c                                  2
      4  tsize      s   3.0000E+00   1.2000E+02
      5  tgrade     c                                  3
      6  pnodes     s   1.0000E+00   5.1000E+01
      7  progrec    s   0.0000E+00   2.3800E+03
      8  estrec     s   0.0000E+00   1.1440E+03
      9  time       t   7.2000E+01   2.6590E+03
     10  death      d   0.0000E+00   1.0000E+00
  =================== Constructed variables ===================
     11  lnbasehaz  z  -6.5103E+00   5.8866E-02
     12  horTh.yes  f   0.0000E+00   1.0000E+00

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       686         0          0        0        0        0        5        0        2
 Survival time variable in column: 9
 Censoring indicator variable in column: 10
 Proportion of uncensored among nonmissing T and D variables: 0.445
 No. cases used for training: 672

 Warning: interaction tests skipped
 Missing values imputed with node means for model fitting
 Treatment interactions (Gi)
 No interaction tests
 Pruning by v-fold cross-validation, with v =  10
 Selected tree is based on mean of CV estimates
 Fraction of cases used for splitting each node =   1.0000
 Max number of split levels =  10
 Minimum node size =  34
 Number of iterations =  5
 Number of SE's for pruned tree =   5.0000E-01

 Size and CV Loss and SE of subtrees:
  Tree   #Tnodes  Mean Loss   SE(Mean)   BSE(Mean)  Median Loss  BSE(Median)
    1       15   1.453E+00   5.891E-02   4.830E-02   1.408E+00   5.079E-02
    2       14   1.453E+00   5.885E-02   4.827E-02   1.408E+00   5.061E-02
    3       13   1.441E+00   5.676E-02   4.165E-02   1.395E+00   5.337E-02
    4       11   1.445E+00   5.640E-02   4.195E-02   1.405E+00   5.353E-02
    5        7   1.444E+00   5.632E-02   4.201E-02   1.405E+00   5.235E-02
    6        5   1.445E+00   5.627E-02   4.149E-02   1.405E+00   5.190E-02
    7**      2   1.408E+00   5.211E-02   3.686E-02   1.386E+00   3.980E-02
    8        1   1.442E+00   5.157E-02   1.216E-02   1.450E+00   1.474E-02

 0-SE tree based on mean is marked with *
 0-SE tree based on median is marked with +
```

```
Selected-SE tree based on mean using naive SE is marked with **
Selected-SE tree based on mean using bootstrap SE is marked with --
Selected-SE tree based on median and bootstrap SE is marked with ++
** tree and ++ tree are the same


Following tree is based on mean CV with naive SE estimate (**).

Structure of final tree. Each terminal node is marked with a T.

Rel. risk is mean risk relative to sample average ignoring covariates

Cases fit give the number of cases used to fit node
Deviance is mean residual deviance for all cases in node
       Node    Total    Cases Matrix    Node       Node    Split
       label   cases      fit  rank   rel.risk   deviance  variable
          1      672      672     1  1.000E+00  1.414E+00  progrec
          2T     274      274     1  1.588E+00  1.584E+00  menostat
          3T     398      398     1  7.095E-01  1.172E+00  menostat

Number of terminal nodes of final tree: 2
Total number of nodes of final tree:    3

Regression tree:

 Node 1: progrec <=   21.00000
   Node 2: Risk relative to sample average ignoring covariates =   1.58824
 Node 1: progrec >   21.00000 or NA
   Node 3: Risk relative to sample average ignoring covariates =   0.70947

***************************************************************

Constant term for constant hazard model (ignoring covariates):  -0.00259998


Node 1: Intermediate node
A case goes into Node 2 if progrec <=  2.1000000E+01
progrec mean =  1.1092E+02
Coefficients of log-relative risk function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant   1.2903E-01        1.85 0.0651
horTh.yes -3.6984E-01       -2.97 0.0031    0.0000E+00  3.6012E-01  1.0000E+00
----------------------------
Node 2: Terminal node
Coefficients of log-relative risk function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
```

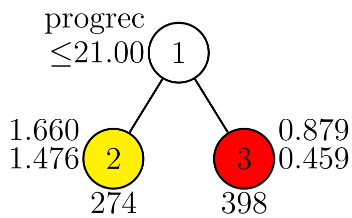

Figure 16: GUIDE Gi proportional hazards regression tree for differential treatment effects At each intermediate node, an observation goes to the left branch if and only if the condition is satisfied. Numbers beside terminal nodes are estimated relative risks (relative to average for sample ignoring covariates) corresponding to treatment levels `no` and `yes`; numbers below are sample sizes.

```
Constant   5.0439E-01        5.04 0.0000
horTh.yes -1.1775E-01       -0.71 0.4786    0.0000E+00  3.6131E-01  1.0000E+00
----------------------------
Node 3: Terminal node
Coefficients of log-relative risk function:
Regressor Coefficient      t-stat  p-val     Minimum        Mean     Maximum
Constant  -1.3184E-01       -1.35 0.1775
horTh.yes -6.5011E-01       -3.40 0.0007    0.0000E+00  3.5930E-01  1.0000E+00
----------------------------

Observed and fitted values are stored in cancer.fit
LaTeX code for tree is in cancer.tex
```

The LaTeX tree diagram and the Kaplan-Meier survival functions estimated from the data in the terminal nodes of the tree are shown in Figures 16 and 17, respectively.

Estimated relative risks and survival probabilities The file `cancer.fit` gives the terminal node number, estimated survival time, log baseline cumulative hazard, relative risk (relative to the average for the data, ignoring covariates), survival probability, and median survival time of each observation in the training sample file `cancer.txt`. The results for the first few observations are shown below. See Section 5.7 for definitions of the terms.

```
train node  survivaltime  logbasecumhaz  relativerisk  survivalprob  mediansurvtime
    y    3  1.814000E+03  -3.317667E-01  8.787636E-01  5.331186E-01    2.014420E+03
    y    3  2.018000E+03  -2.024282E-01  4.587030E-01  6.882035E-01    2.659000E+03+
    y    3  7.120000E+02  -1.300331E+00  4.587030E-01  8.828100E-01    2.659000E+03+
    y    3  1.807000E+03  -3.550694E-01  4.587030E-01  7.255880E-01    2.659000E+03+
    y    3  7.720000E+02  -1.176558E+00  8.787636E-01  7.631865E-01    2.014420E+03
```

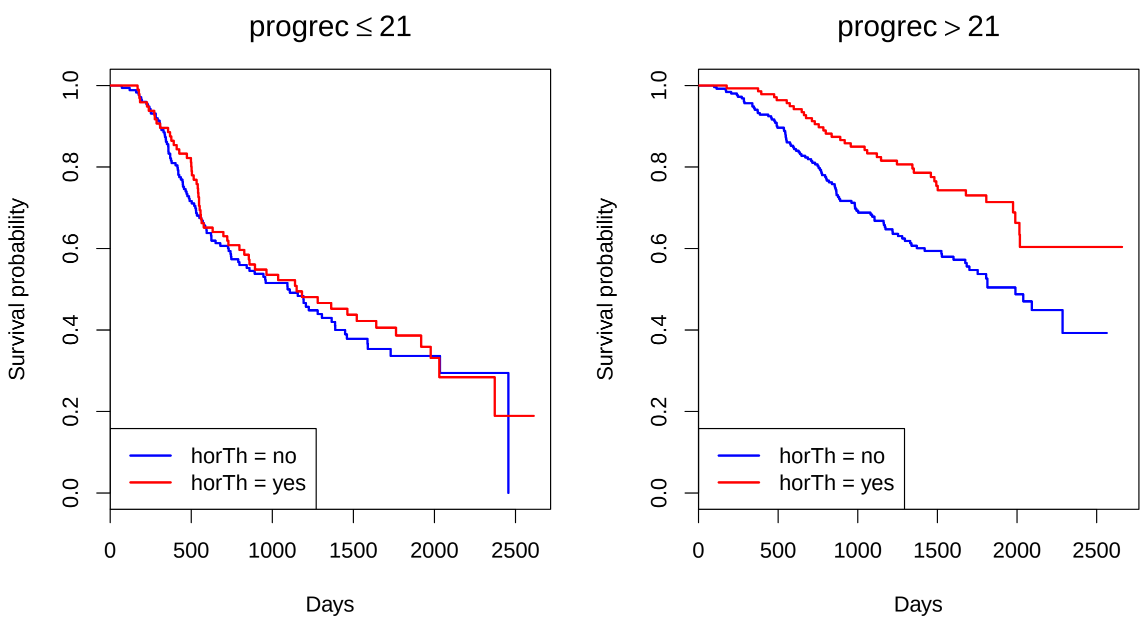

Figure 17: Estimated survival probability functions for breast cancer data

```
y    2  4.480000E+02  -2.105688E+00  1.660293E+00  8.173929E-01    1.038277E+03
```

5.11 Differential item functioning

GUIDE has an experimental option to identify important predictor variables and items with differential item functioning (DIF) in a data set with two or more item (dependent variable) scores. We illustrate it with a data set from Broekman et al. (2011, 2008) and Marc et al. (2008). It consists of responses from 1978 subjects on 15 items. There are 3 predictor variables (age, education, and gender). The data and description files are `GDS.dat` and `GDS.dsc`. Although the item responses in this example are 0-1, GUIDE allows them to be in any ordinal (e.g., Likert) scale. The contents of `GDS.dsc` are:

```
GDS.dat
NA
1
1 SATIS d
2 SPIRIT d
3 HAPPY d
4 ALIVE d
5 ENERGY d
6 DROP d
```

```
7 EMPTY d
8 BORED d
9 AFRAID d
10 HELP d
11 HOME d
12 MEMORY d
13 WORTH d
14 HOPE d
15 BETTER d
16 MALE c
17 EDUCATION n
18 AGE n
```

Here is the session log to create an input file for identifying DIF items and the important predictor variables:

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning without input file creation
Input your choice: 1
Name of batch input file: GDSimp.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: GDSimp.out
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
Choose type of regression model:
 1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
 5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
Input choice ([1:6], <cr>=1): 5
 Choose option 5 for item response data.
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: GDS.dsc
Reading data description file ...
Training sample file: GDS.dat
Missing value code: NA
Records in data file start on line 1
Warning: N variables changed to S
Number of D variables = 15
D variables are:
SATIS
SPIRIT
HAPPY
ALIVE
```

```
ENERGY
DROP
EMPTY
BORED
AFRAID
HELP
HOME
MEMORY
WORTH
HOPE
BETTER
Choose multivariate or univariate split variable selection:
Choose multivariate if there is an order among the D variables; otherwise choose univariate
Input 1 for multivariate, 2 for univariate ([1:2], <cr>=2):
Input 1 to normalize D variables, 2 for no normalization ([1:2], <cr>=2):
Input 1 for equal, 2 for unequal weighting of D variables ([1:2], <cr>=1):
Reading data file ...
Number of records in data file: 1978
Length of longest data entry: 2
Checking for missing values ...
Total number of cases: 1978
Col. no. Categorical variable    #levels    #missing values
      16 MALE                          2                  0
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Rereading data
PCA can be used for variable selection
Do not use PCA if differential item functioning (DIF) scores are wanted
Input 1 to use PCA, 2 otherwise ([1:2], <cr>=2):
 Choose the default because DIF scoring is desired.

#cases w/ miss. D = number of cases with all D values missing
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
      1978          0          0        0        0        0        2        0        1
No. cases used for training: 1978
Finish reading data file
Warning: interaction tests skipped
Input expected fraction of noise variables erroneously selected ([0.00:0.99], <cr>=0.01):
Input 1 to save p-value matrix for differential item functioning (DIF), 2 otherwise ([1:2], <cr>=1
Input file name to store DIF p-values: GDSimp.pv
A file by that name already exists
Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
You can create a description file with the selected variables included or excluded
Input 2 to create such a file, 1 otherwise ([1:2], <cr>=1): 2
```

```
Input 1 to keep only selected variables, 2 to exclude selected variables ([1:2], <cr>=1):
Input file name: GDSsub.dsc
File GDSsub.dsc exists
Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
You can also output the importance scores and variable names to a file
Input 1 to create such a file, 2 otherwise ([1:2], <cr>=1):
Input file name: GDSimp.scr
A file by that name already exists
Input 1 to overwrite it, 2 to choose another name ([1:2], <cr>=1):
Input file is created!
Run GUIDE with the command: guide < GDSimp.in
```

The importance scores in the output file `GDSimp.scr` shows that all three predictor variables are distinguishable from noise (because their scores are above 1.0):

```
      Rank        Score    Variable
         1        7.519    AGE
         2        3.501    MALE
         3        1.598    EDUCATION
```

The last column of `GDSimp.pv` below shows that three items (#4, 10, 13) have DIF.

```
Item     Itemname    EDUCATION          AGE         MALE   DIF
   1        SATIS    0.310E-01    0.946E-01    0.476E-01   no
   2       SPIRIT    0.988E+00    0.456E+00    0.437E-02   no
   3        HAPPY    0.938E+00    0.930E-01    0.375E-01   no
   4        ALIVE    0.129E+00    0.282E-01    0.382E+00   no
   5       ENERGY    0.721E+00    0.845E+00    0.573E-06   yes
   6         DROP    0.107E-01    0.117E+00    0.951E+00   no
   7        EMPTY    0.369E-02    0.194E-02    0.315E-01   no
   8        BORED    0.750E-07    0.166E+00    0.416E+00   yes
   9       AFRAID    0.106E+00    0.323E-02    0.287E-02   no
  10         HELP    0.928E-01    0.678E+00    0.148E-02   no
  11         HOME    0.128E+00    0.826E+00    0.779E-03   no
  12       MEMORY    0.434E+00    0.000E+00    0.440E-01   yes
  13        WORTH    0.934E+00    0.573E+00    0.624E+00   no
  14         HOPE    0.653E+00    0.799E+00    0.109E+00   no
  15       BETTER    0.956E+00    0.525E+00    0.747E+00   no
```

The following output file `GDSsub.dsc` can be used to fit a model to the selected item and predictor variables:

```
"GDS.dat"
```

```
"NA"
1
1 SATIS x
2 SPIRIT x
3 HAPPY x
4 ALIVE x
5 ENERGY d
6 DROP x
7 EMPTY x
8 BORED d
9 AFRAID x
10 HELP x
11 HOME x
12 MEMORY d
13 WORTH x
14 HOPE x
15 BETTER x
16 MALE c
17 EDUCATION n
18 AGE n
```

Following is the input file creation log that uses `GDSsub.dsc`.

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning without input file creation
Input your choice: 1
Name of batch input file: GDSsub.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: GDSsub.out
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
Choose type of regression model:
 1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
 5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
Input choice ([1:6], <cr>=1): 5
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: GDSsub.dsc
Reading data description file ...
Training sample file: GDS.dat
Missing value code: NA
Records in data file start on line 1
```

```
Warning: N variables changed to S
Number of D variables = 3
D variables are:
ENERGY
BORED
MEMORY
Choose multivariate or univariate split variable selection:
Choose multivariate if there is an order among the D variables; otherwise choose univariate
Input 1 for multivariate, 2 for univariate ([1:2], <cr>=1): 2
 Choose 2 because items are not ordered.
Input 1 to normalize D variables, 2 for no normalization ([1:2], <cr>=2):
Input 1 for equal, 2 for unequal weighting of D variables ([1:2], <cr>=1):
Reading data file ...
Number of records in data file: 1978
Length of longest data entry: 2
Checking for missing values ...
Total number of cases: 1978
Col. no. Categorical variable    #levels    #missing values
      16 MALE                          2                  0
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Rereading data
PCA can be used for variable selection
Do not use PCA if differential item functioning (DIF) scores are wanted
Input 1 to use PCA, 2 otherwise ([1:2], <cr>=2):
#cases w/ miss. D = number of cases with all D values missing
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
      1978          0          0       12        0        0        2        0        1
No. cases used for training: 1978
Finish reading data file
Warning: interaction tests skipped
Input 1 for LaTeX tree code, 2 to skip it ([1:2], <cr>=1):
Input file name to store LaTeX code (use .tex as suffix): GDSsub.tex
Input 2 to save node IDs of individual cases, 1 otherwise ([1:2], <cr>=2):
Input name of file to store terminal node ID of each case: GDSsub.nid
Input 2 to save fitted values at each terminal node; 1 otherwise ([1:2], <cr>=1): 2
Input name of file to store node fitted values: GDSsub.fit
Input file is created!
Run GUIDE with the command: guide < GDSsub.in
```

The result of running this input file produces a tree with four terminal nodes shown in Figure 18 with mean item responses printed below the terminal nodes. The file `GDSsub.nid` gives the terminal node number that each case resides in the tree. The first 10 casea are shown below.

```
case train         node
   1     y           15
   2     y           15
   3     y           14
   4     y           15
   5     y            6
   6     y           15
   7     y            6
   8     y            6
   9     y            6
  10     y           15
```

The file `GDSsub.fit` gives the sample mean values of the dependent (item) variables in each terminal node:

```
node   ENERGY       BORED        MEMORY
   2   0.24345E+00  0.13109E+00  0.64419E+00
   6   0.26058E+00  0.62169E-01  0.34259E+00
  14   0.12179E+00  0.76923E-01  0.50641E+00
  15   0.21151E+00  0.76345E-01  0.38924E+00
```

6 Tree ensembles

A tree ensemble is a collection of trees. GUIDE has two methods of constructing an ensemble. One is called "bagged GUIDE", which fits *pruned* GUIDE trees to bootstrap samples of the training data (Breiman, 1996). The other is called "GUIDE forest"; it is similar to random forest (Breiman, 2001), which fits *unpruned* trees to bootstrap samples but randomly selects a small subset of variables for split selection at each node. There is some empirical evidence that, if there are many variables of which only a few are useful for prediction, bagged GUIDE tends to be more accurate than GUIDE forest (Loh, 2009, 2012). But GUIDE forest is computationally faster.

6.1 Bagged GUIDE

We first demonstrate bagged GUIDE on the car data.

```
 0. Read the warranty disclaimer
 1. Create an input file for batch run
 2. Fit a model without creating input file
 3. Convert data to other formats
```

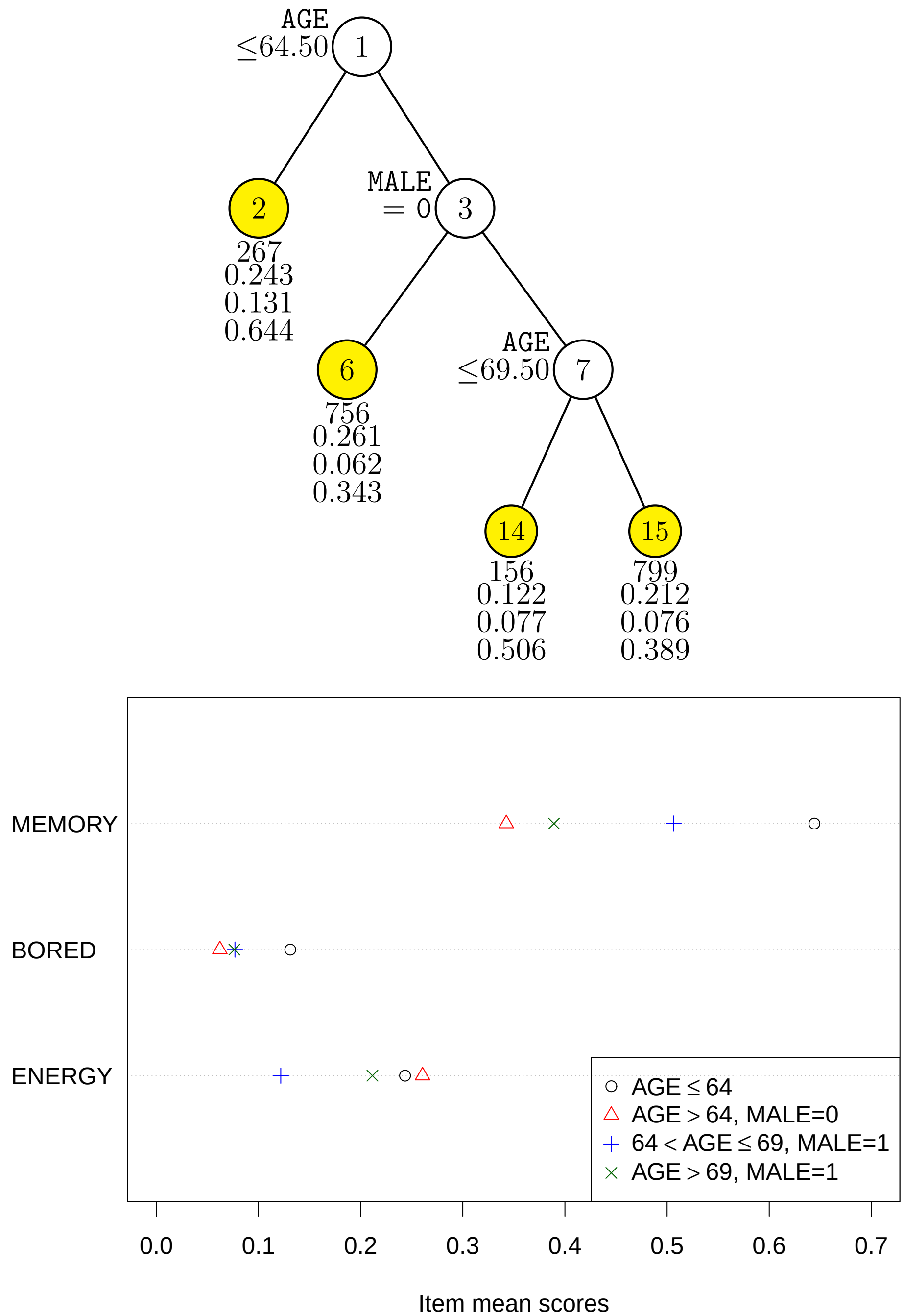

Figure 18: GUIDE 0.50-SE regression tree for predicting response variables `ENERGY`, `BORED`, and `MEMORY`. PCA not used. At each split, an observation goes to the left branch if and only if the condition is satisfied. Sample sizes and predicted values of `ENERGY`, `BORED`, and `MEMORY` are printed below nodes.

```
4. Variable importance scoring and differential item functioning without input file creation
Input your choice: 1
Name of batch input file: bagin.txt
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: bagout.txt
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1): 2
 This is where an ensemble method is selected.
Input 1 for bagging, 2 for rforest: ([1:2], <cr>=2): 1
 Option 1 is bagged GUIDE, option 2 is GUIDE forest.
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: drivedsc.txt
Reading data description file ...
Training sample file: drive.txt
Missing value code: *
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is Drive
Reading data file ...
Number of records in data file: 428
Length of longest data entry: 26
Checking for missing values ...
Total number of cases: 428
Number of classes =            3
Col. no. Categorical variable    #levels    #missing values
       3 Make                         38                  0
       5 Type                          6                  0
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Rereading data
Class      #Cases    Proportion
4wd            94    0.21962617
fwd           224    0.52336449
rwd           110    0.25700935
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals    #X-var    #N-var    #F-var    #S-var    #B-var    #C-var
       428          0          0        10         0         0        11         0         2
No. cases used for training: 428
Finish reading data file
Choose 1 for estimated priors, 2 for equal priors, 3 for priors from a file
Input 1, 2, or 3 ([1:3], <cr>=1):
Choose 1 for unit misclassification costs, 2 to input costs from a file
Input 1 or 2 ([1:2], <cr>=1):
```

```
Input name of file to store predicted class and probability: bagfit.txt
Input file is created!
Run GUIDE with the command: guide < bagin.txt
```

Results

```
Ensemble of bagged classification trees
Pruning by cross-validation
Data description file: drivedsc.txt
Training sample file: drive.txt
Missing value code: *
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is Drive
Number of records in data file: 428
Length of longest data entry: 26
Number of classes =  3
Class      #Cases    Proportion
4wd            94    0.21962617
fwd           224    0.52336449
rwd           110    0.25700935

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
 Column  Name            Minimum      Maximum   #Categories       #Missing
      3  Make       c                                    38
      5  Type       c                                     6
      6  Drive      d                                     3
     14  Rprice     s   1.0280E+04   1.9246E+05
     15  Dcost      s   9.8750E+03   1.7356E+05
     16  Enginsz    s   1.3000E+00   8.3000E+00
     17  Cylin      s  -1.0000E+00   1.2000E+01
     18  Hp         s   7.3000E+01   5.0000E+02
     19  City       s   1.0000E+01   6.0000E+01
     20  Hwy        s   1.2000E+01   6.6000E+01
     21  Weight     s   1.8500E+03   7.1900E+03
     22  Whlbase    s   8.9000E+01   1.4400E+02
     23  Length     s   1.4300E+02   2.2800E+02
     24  Width      s   6.4000E+01   8.1000E+01

     Total  #cases w/   #missing
    #cases    miss. D  ord. vals    #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       428          0          0        10        0        0       11        0        2
No. cases used for training: 428
```

```
 Univariate split highest priority
 Interaction splits 2nd priority; no linear splits
 Number of trees in ensemble =  100
 Pruning by v-fold cross-validation, with v =  5
 Selected tree is based on mean of CV estimates
 Simple node models
 Estimated priors
 Unit misclassification costs
 Fraction of cases used for splitting each node =  0.23364
 Max number of split levels =  7
 Minimum node size =  10
 Number of SE's for pruned tree =   5.0000E-01

 Mean number of terminal nodes =     8.010

 Classification matrix for training sample:
 Predicted       True class
class            4wd       fwd       rwd
 4wd               52        16         6
 fwd               26       203        15
 rwd               16         5        89
 Total             94       224       110

 Number of cases used for tree construction =  428
 Number misclassified =  84
 Resubstitution est. of mean misclassification cost =   0.19626168224299062
 Note: The above results will likely differ slightly from one run to another
 due to the randomness of bagging.
 Predicted class probability estimates are stored in bagfit.txt
```

The following lines from the top of the file `bagfit.txt` give the estimated class probabilities and the predicted class of the observations.

```
 "4wd" "fwd" "rwd" predicted observed
  0.48170E-01  0.90727E+00  0.44555E-01    "fwd"    "fwd"
  0.52722E-01  0.90205E+00  0.45229E-01    "fwd"    "fwd"
  0.53247E-01  0.90195E+00  0.44808E-01    "fwd"    "fwd"
  0.50013E-01  0.90280E+00  0.47186E-01    "fwd"    "fwd"
  0.55602E-01  0.89210E+00  0.52298E-01    "fwd"    "fwd"
  0.78480E-01  0.83218E+00  0.89341E-01    "fwd"    "fwd"
  0.59715E-01  0.84087E+00  0.99414E-01    "fwd"    "fwd"
  0.61966E-01  0.86091E+00  0.77127E-01    "fwd"    "fwd"
```

6.2 GUIDE forest

GUIDE forest differs from bagged GUIDE in two respects:

1. At each node a random subset of the variables is used for split selection.
2. The trees in GUIDE forest are not pruned.

These are the same principles in random forest. GUIDE forest differs from the latter in using GUIDE's unbiased variable selection method instead of greedy search. GUIDE forest typically requires many more trees than bagged GUIDE to achieve similar accuracy, but because the former does not prune the trees, the former is still faster to compute.

Input file creation

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning without input file creation
Input your choice: 1
Name of batch input file: forestin.txt
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
Name of batch output file: forestout.txt
Input 1 for single tree, 2 for ensemble ([1:2], <cr>=1): 2
Input 1 for bagging, 2 for rforest: ([1:2], <cr>=2):
Input 1 for random splits of missing values, 2 for nonrandom: ([1:2], <cr>=2):
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):

Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: drivedsc.txt
Reading data description file ...
Training sample file: drive.txt
Missing value code: *
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is Drive
Reading data file ...
Number of records in data file: 428
Length of longest data entry: 26
Checking for missing values ...
Total number of cases: 428
Number of classes =           3
```

```
Col. no. Categorical variable    #levels    #missing values
       3 Make                         38                  0
       5 Type                          6                  0
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Rereading data
Class      #Cases    Proportion
4wd            94    0.21962617
fwd           224    0.52336449
rwd           110    0.25700935
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       428          0          0       10        0        0       11        0        2
No. cases used for training: 428
Finish reading data file
Choose 1 for estimated priors, 2 for equal priors, 3 for priors from a file
Input 1, 2, or 3 ([1:3], <cr>=1):
Choose 1 for unit misclassification costs, 2 to input costs from a file
Input 1 or 2 ([1:2], <cr>=1):
Input name of file to store predicted class and probability: forest.fit
Input file is created!
Run GUIDE with the command: guide < forestin.txt
```

Results

```
Random forest of classification trees
No pruning
Data description file: drivedsc.txt
Training sample file: drive.txt
Missing value code: *
Records in data file start on line 1
Warning: N variables changed to S
Dependent variable is Drive
Number of records in data file: 428
Length of longest data entry: 26
Number of classes =  3
Class      #Cases    Proportion
4wd            94    0.21962617
fwd           224    0.52336449
rwd           110    0.25700935

Summary information (without x variables)
d=dependent, b=split and fit cat variable using 0-1 dummies, c=split-only categorical,
n=split and fit numerical, f=fit-only numerical, s=split-only numerical, w=weight
```

```
  Column  Name            Minimum      Maximum   #Categories       #Missing
       3  Make       c                                    38
       5  Type       c                                     6
       6  Drive      d                                     3
      14  Rprice     s   1.0280E+04   1.9246E+05
      15  Dcost      s   9.8750E+03   1.7356E+05
      16  Enginsz    s   1.3000E+00   8.3000E+00
      17  Cylin      s  -1.0000E+00   1.2000E+01
      18  Hp         s   7.3000E+01   5.0000E+02
      19  City       s   1.0000E+01   6.0000E+01
      20  Hwy        s   1.2000E+01   6.6000E+01
      21  Weight     s   1.8500E+03   7.1900E+03
      22  Whlbase    s   8.9000E+01   1.4400E+02
      23  Length     s   1.4300E+02   2.2800E+02
      24  Width      s   6.4000E+01   8.1000E+01

      Total  #cases w/   #missing
     #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
        428          0          0       10        0        0       11        0        2
 No. cases used for training: 428

 Univariate split highest priority
 No interaction and linear splits
 Number of trees in ensemble =  500
 Number of variables used for splitting =  5
 Simple node models
 Estimated priors
 Unit misclassification costs
 Fraction of cases used for splitting each node =  0.23364
 Max number of split levels =  10
 Minimum node size =  5
 Mean number of terminal nodes =     29.76

 Classification matrix for training sample:
 Predicted       True class
class           4wd       fwd       rwd
 4wd              58        17         7
 fwd              20       200         7
 rwd              16         7        96
 Total            94       224       110

 Number of cases used for tree construction =  428
 Number misclassified =  74
 Resubstitution est. of mean misclassification cost =   0.17289719626168223

 Predicted class probability estimates are stored in forest.fit
```

The above results are not particularly useful because it is impossible to analyze the individual trees. The results mostly provide a record of the parameter values chosen to construct the forest. The most interesting results are the predicted values in the file `forest.fit`, the top few lines of which are shown below.

```
"4wd" "fwd" "rwd" predicted observed
 0.96084E-02  0.97863E+00  0.11763E-01    "fwd"    "fwd"
 0.10052E-01  0.97323E+00  0.16714E-01    "fwd"    "fwd"
 0.41149E-02  0.98654E+00  0.93414E-02    "fwd"    "fwd"
 0.34438E-02  0.99011E+00  0.64457E-02    "fwd"    "fwd"
 0.11450E-01  0.97050E+00  0.18051E-01    "fwd"    "fwd"
 0.21658E-01  0.93736E+00  0.40979E-01    "fwd"    "fwd"
 0.26231E-01  0.93399E+00  0.39777E-01    "fwd"    "fwd"
```

7 Importance scores

GUIDE can rank the variables in order of their importance for predicting the dependent variable. In addition, it provides a threshold score for distinguishing the important variables from the unimportant ones.

7.1 Baseball data example

We demonstrate this capability with the baseball data below.

```
0. Read the warranty disclaimer
1. Create an input file for batch run
2. Fit a model without creating input file
3. Convert data to other formats
4. Variable importance scoring and differential item functioning without input file creation
Input your choice: 1
Name of batch input file: bbimp.in
Input 1 for model fitting, 2 for importance or DIF scoring, 3 for data conversion ([1:3], <cr>=1):
 Option 2 yields importance scores.
Name of batch output file: bbimp.out
Input 1 for classification, 2 for regression, 3 for propensity score grouping ([1:3], <cr>=1): 2
Choose type of regression model:
 1=linear, 2=quantile, 3=Poisson, 4=proportional hazards,
 5=multiresponse or itemresponse, 6=longitudinal data (with T variables).
Input choice ([1:6], <cr>=1):
Input 1 for least squares, 2 least median of squares ([1:2], <cr>=1):
Input 1 for default options, 2 otherwise ([1:2], <cr>=1):
```

```
Input name of data description file (max 100 characters);
enclose with matching quotes if it has spaces: bbdsc.txt
Reading data description file ...
Training sample file: bbdat.txt
Missing value code: NA
Records in data file start on line 1
Warning: N variables changed to S
Warning: B variables changed to C
Dependent variable is Logsalary
Reading data file ...
Number of records in data file: 263
Length of longest data entry: 17
Checking for missing values ...
Total number of cases: 263
Col. no. Categorical variable    #levels    #missing values
      16 Leag86                        2                  0
      17 Div86                         2                  0
      18 Team86                       24                  0
      19 Pos86                        23                  0
      24 Leag87                        2                  0
      25 Team87                       24                  0
Re-checking data ...
Assigning codes to categorical and missing values
Finish checking data
Rereading data
     Total  #cases w/   #missing
    #cases    miss. D  ord. vals   #X-var   #N-var   #F-var   #S-var   #B-var   #C-var
       263          0          0        3        0        0       16        0        6
No weight variable in data file
No. cases used for training: 263
Finish reading data file
Input expected fraction of noise variables erroneously selected ([0.00:0.99], <cr>=0.01):
 This sets the ``alpha'' value such that, under the null hypothesis that all
 variables are noise, the proportion erroneously selected is alpha.
You can create a description file with the selected variables included or excluded
Input 2 to create such a file, 1 otherwise ([1:2], <cr>=1): 2
 This option lets GUIDE automatically write a new description file with the
 unimportant variables given the X designation.
Input 1 to keep only selected variables, 2 to exclude selected variables ([1:2], <cr>=1):
Input file name: bbsub.dsc
You can also output the importance scores and variable names to a file
Input 1 to create such a file, 2 otherwise ([1:2], <cr>=1):
Input file name: bbimp.scr
Input file is created!
Run GUIDE with the command: guide < bbimp.in
```

Results The importance scores are given at the end of the output file `bbimp.out`.

```
 Predictor variables sorted by importance scores
 Importance Scores
 100.0    1.57802E+01      1    Hitcr
  94.7    1.49442E+01      2    Batcr
  86.8    1.36945E+01      3    Runcr
  82.6    1.30399E+01      4    Rbcr
  75.0    1.18373E+01      5    Yrs
  68.2    1.07689E+01      6    Wlkcr
  47.8    7.55054E+00      7    Hrcr
  47.0    7.41739E+00      8    Hit86
  36.8    5.79925E+00      9    Run86
  35.4    5.59201E+00     10    Rb86
  35.3    5.56333E+00     11    Bat86
  31.7    5.00925E+00     12    Wlk86
  20.4    3.21772E+00     13    Hr86
  10.8    1.70322E+00     14    Pos86
   8.9    1.40858E+00     15    Puto86
   6.5    1.02078E+00     16    Team87
----------------- cut-off -------------------
   4.5    7.07834E-01     17    Err86
   4.1    6.53864E-01     18    Asst86
   2.8    4.43732E-01     19    Team86
   2.3    3.61595E-01     20    Leag87
   1.8    2.77317E-01     21    Leag86
   1.6    2.51640E-01     22    Div86
Variables with unscaled scores above 1 are considered important

Number of important and unimportant split variables: 16, 6
Importance scores are stored in bbimp.scr
Description file with selected variables in bbsub.dsc
```

Here are the contents of the file `bbimp.scr`:

```
     Rank        Score    Variable
        1       15.780    Hitcr
        2       14.944    Batcr
        3       13.695    Runcr
        4       13.040    Rbcr
        5       11.837    Yrs
        6       10.769    Wlkcr
        7        7.551    Hrcr
        8        7.417    Hit86
        9        5.799    Run86
       10        5.592    Rb86
```

```
       11       5.563    Bat86
       12       5.009    Wlk86
       13       3.218    Hr86
       14       1.703    Pos86
       15       1.409    Puto86
       16       1.021    Team87
       17       0.708    Err86
       18       0.654    Asst86
       19       0.444    Team86
       20       0.362    Leag87
       21       0.277    Leag86
       22       0.252    Div86
```

And here are the contents of the file `bbsub.dsc`:

```
"bbdat.txt"
"NA"
1
1 Id x
2 Name x
3 Bat86 n
4 Hit86 n
5 Hr86 n
6 Run86 n
7 Rb86 n
8 Wlk86 n
9 Yrs n
10 Batcr n
11 Hitcr n
12 Hrcr n
13 Runcr n
14 Rbcr n
15 Wlkcr n
16 Leag86 x
17 Div86 x
18 Team86 x
19 Pos86 c
20 Puto86 n
21 Asst86 x
22 Err86 x
23 Salary x
24 Leag87 x
25 Team87 c
26 Logsalary d
```

8 Other features

8.1 Pruning with test samples

GUIDE typically has three pruning options for deciding the size of the final tree: (i) cross-validation, (ii) test sample, and (iii) no pruning. Test-sample pruning is available only when there are no derived variables, such as creation of dummy indicator variables when '`b`' variables are present. If test-sample pruning is chosen, the program will ask for the name of the file containing the test samples. This file must have the same column format as the training sample file. Pruning with test-samples or no pruning are non-default options.

8.2 Prediction of test samples

GUIDE can produce R code to predict future observations from all except kernel and nearest neighbor classification and ensemble models. This is also a non-default option.

Predictions of the training data for all models can be obtained, however, at the time of tree construction. This feature can be used to obtain predictions on "test samples" (i.e., observations that are not used in tree construction) by adding them to the training sample file. There are two ways to distinguish the test observations from the training observations:

1. Use a *weight* variable (designated as `W` in the description file) that takes value 1 for each training observation and 0 or each test observation.

2. Replace the `D` values of the test observations with the missing value code.

For tree construction, GUIDE does not use observations in the training sample file that have zero weight.

8.3 GUIDE in R and in simulations

GUIDE can be used in simulations or used repeatedly on bootstrap samples to produce an ensemble of tree models. For the latter,

1. Create a file (with name `data.txt`, say) containing one set of bootstrapped data.

2. Create a data description file (with name `desc.txt`, say) that refers to `data.txt`.

3. Create an input file (with name `input.txt`, say) that refers to `desc.txt`.
4. Write a batch program (Windows) or a shell script (Linux or Macintosh) that repeatedly:
   (a) replaces the file `data.txt` with new bootstrapped samples;
   (b) calls GUIDE with the command: `guide < input.txt`; and
   (c) reads and processes the results from each GUIDE run.

   In R, the command in step 4b depends on the operating system. If the GUIDE program and the files `data.txt` and `input.txt` are in the same folder as the working R directory, the command is:

   Linux/Macintosh: `system("guide < input.txt > log.txt")`

   Windows: `shell("guide < input.txt > log.txt")`

   If the files are not all in the same folder, full path names must be given. Here `log.txt` is a text file that stores messages during execution. If GUIDE does not run successfully, errors are also written to `log.txt`.

8.4 Generation of powers and products

GUIDE allows the creation of certain powers and products of regressor variables on the fly. Specifically, variables of the form $X_1^p X_2^q$, where X_1 and X_2 are numerical predictor variables and p and q are integers, can be created by adding one or more lines of the form

```
0 i p j q a
```

at the end of the data description file. Here `i` and `j` are integers giving the column numbers of variables X_1 and X_2, respectively, in the data file and `a` is one of the letters `n`, `s`, or `f` (corresponding to a numerical variable used for both splitting and fitting, splitting only, or fitting only).

To illustrate, suppose we wish to fit a piecewise quadratic model in the variable `Yrs` for the baseball data. This is easily done by adding one line to the file `bbdsc.txt`. First we assign the `s` (for splitting only) designator to every numerical predictor except `Yrs`. This will prevent all variables other than `Yrs` from acting as regressors in the piecewise quadratic models. To create the variable `Yrs`^{2}, add the line

```
0 9 2 9 0 f
```

to the end of `bbdsc.txt`. The `9`'s in the above line refers to the column number of the variables `Yrs` in the data file, and the `f` tells the program to use the variable $\texttt{Yrs}^2$ for fitting terminal node models only. Note: The line defines $\texttt{Yrs}^2$ as $\texttt{Yrs}^2 \times \texttt{Yrs}^0$. Since we can equivalently define the variable by $\texttt{Yrs}^2 = \texttt{Yrs}^1 \times \texttt{Yrs}^1$, we could also have used the line: "`0 9 1 9 1 f`".

The resulting description file now looks like this:

```
bbdat.txt
NA
1
1 Id x
2 Name x
3 Bat86 s
4 Hit86 s
5 Hr86 s
6 Run86 s
7 Rb86 s
8 Wlk86 s
9 Yrs n
10 Batcr s
11 Hitcr s
12 Hrcr s
13 Runcr s
14 Rbcr s
15 Wlkcr s
16 Leag86 c
17 Div86 c
18 Team86 c
19 Pos86 c
20 Puto86 s
21 Asst86 s
22 Err86 s
23 Salary x
24 Leag87 c
25 Team87 c
26 Logsalary d
0 9 2 9 0 f
```

When the program is given this description file, the output will show the regression coefficients of `Yrs` and $\texttt{Yrs}^2$ in each terminal node of the tree.

8.5 Data formatting functions

The program includes a utility function for reformatting data files into forms required by some statistical software packages:

1. R/Splus: Fields are space delimited. Missing values are coded as `NA`. Each record is written on one line. Variable names are given on the first line.

2. SAS: Fields are space delimited. Missing values are coded with periods. Character strings are truncated to eight characters. Spaces within character strings are replaced with underscores (_).

3. TEXT: Fields are comma delimited. Empty fields denote missing values. Character strings longer than eight characters are truncated. Each record is written on one line. Variable names are given on the first line.

4. STATISTICA: Fields are comma delimited. Commas in character strings are stripped. Empty fields denote missing values. Each record occupies one line.

5. SYSTAT: Fields are comma delimited. Strings are truncated to eight characters. Missing character values are replaced with spaces, missing numerical values with periods. Each record occupies one line.

6. BMDP: Fields are space delimited. Categorical values are sorted in alphabetic order and then assigned integer codes. Missing values are indicated by asterisks. Variable names longer than eight characters are truncated.

7. DataDesk: Fields are space delimited. Missing categorical values are coded with question marks. Missing numerical values are coded with asterisks. Each record is written on one line. Spaces within categorical values are replaced with underscores. Variable names are given on the first line of the file.

8. MINITAB: Fields are space delimited. Categorical values are sorted in alphabetic order and then assigned integer codes. Missing values are coded with asterisks. Variable names longer than eight characters are truncated.

9. NUMBERS: Same as **TEXT** option except that categorical values are converted to integer codes.

10. C4.5: This is the format required by the C4.5 (Quinlan, 1993) program.

11. ARFF: This is the format required by the WEKA (Witten and Frank, 2000) programs.

Following is a sample session where the iris data are reformatted for R or Splus.

```
 0. Read the warranty disclaimer
 1. Create an input file for batch run
 2. Fit a model without creating input file
 3. Convert data to other formats
 4. Variable importance scoring and differential item functioning
 Input your choice: 3
 Input name of log file: log.txt
  Input 1 if D variable is categorical, 2 if real, 0 if none ([0:2], <cr>=1):

 Input name of data description file (maximum 100 characters; enclose within quotes
 if it contains spaces or non alphanumeric characters): irisdsc.txt
 Reading data description file ...
 Training sample file: irisdata.txt
 Missing value code: ?
 Warning: N variables changed to S
 Dependent variable is class
 Length of longest data entry = 11
 Total number of cases =        150
 Number of classes =           3
 Choose one of the following data formats:
              Field  Miss.val.codes
No. Name       Separ  char.   numer. Remarks
------------------------------------------------------------------
 1  R/Splus    space  NA      NA     1 line/case, var names on 1st line
 2  SAS        space  .       .      strings trunc., spaces -> '_'
 3  TEXT       comma  empty   empty  1 line/case, var names on 1st line
 4  STATISTICA comma  empty   empty  1 line/case, commas stripped
                                     var names on 1st line
 5  SYSTAT     comma  space   .      1 line/case, var names on 1st line
                                     strings trunc. to 8 chars
 6  BMDP       space          *      strings trunc. to 8 chars
                                     cat values -> integers (alph. order)
 7  DATADESK   space  ?       *      1 line/case, var names on 1st line
                                     spaces -> '_'
 8  MINITAB    space          *      cat values -> integers (alph. order)
                                     var names trunc. to 8 chars
 9  NUMBERS    comma  NA      NA     1 line/case, var names on 1st line
                                     cat values -> integers (alph. order)
10  C4.5       comma  ?       ?      1 line/case, dependent variable last
11  ARFF       comma  ?       ?      1 line/case
------------------------------------------------------------------
 0                                   abort this job
 Input your choice ([0:11], <cr>=1):
 Input name of new data file: iris.rdata
 Follow the commented lines in "iris.rdata" to read the data into R or Splus
```

References

Andrews, D. and Herzberg, A. (1985). *Data: a collection of problems from many fields for the student and research worker*. Springer.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24:123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, 45:5–32.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth, Belmont.

Broekman, B. F. P., Niti, M., Nyunt, M. S. Z., Ko, S. M., Kumar, R., and Ng, T. P. (2011). Validation of a brief seven-item response bias-free Geriatric Depression Scale. *American Journal of Geriatric Psychiatry*, 19:589–596.

Broekman, B. F. P., Nyunt, S. Z., Niti, M., Jin, A. Z., Ko, S. M., Kumar, R., Fones, C. S. L., and Ng, T. P. (2008). Differential item functioning of the Geriatic Depression Scale in an Asian population. *Journal of Affective Disorders*, 108:285–290.

Chan, K.-Y. and Loh, W.-Y. (2004). LOTUS: An algorithm for building accurate and comprehensible logistic regression trees. *Journal of Computational and Graphical Statistics*, 13:826–852.

Chaudhuri, P. and Loh, W.-Y. (2002). Nonparametric estimation of conditional quantiles using quantile regression trees. *Bernoulli*, 8:561–576. www.stat.wisc.edu/~loh/treeprogs/guide/quantile.pdf.

Hallin, M. and Ingenbleek, J.-F. (1983). The Swedish automobile portfolio in 1977. *Scandinavian Actuarial Journal*, 83:49–64.

Hoaglin, D. C. and Velleman, P. F. (1995). A critical look at some analyses of major league baseball salaries. *American Statistician*, 49:277–285.

Hosmer, D. W., Lemeshow, S., and May, S. (2008). *Applied Survival Analysis*. Wiley, 2nd edition.

Kim, H. and Loh, W.-Y. (2001). Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96:589–604. www.stat.wisc.edu/~loh/treeprogs/cruise/cruise.pdf.

Kim, H. and Loh, W.-Y. (2003). Classification trees with bivariate linear discriminant node models. *Journal of Computational and Graphical Statistics*, 12:512–530. www.stat.wisc.edu/~loh/treeprogs/cruise/jcgs.pdf.

Kim, H., Loh, W.-Y., Shih, Y.-S., and Chaudhuri, P. (2007). Visualizable and interpretable regression models with good prediction power. *IIE Transactions*, 39:565–579. www.stat.wisc.edu/~loh/treeprogs/guide/iie.pdf.

Koenker, R. W. and Bassett, G. (1978). Regression quantiles. *Econometrica*, 46:33–50.

Loh, W.-Y. (2002). Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12:361–386. www3.stat.sinica.edu.tw/statistica/j12n2/j12n21/j12n21.htm.

Loh, W.-Y. (2006a). Logistic regression tree analysis. In Pham, H., editor, *Handbook of Engineering Statistics*, pages 537–549. Springer.

Loh, W.-Y. (2006b). Regression tree models for designed experiments. In Rojo, J., editor, *The Second Erich L. Lehmann Symposium–Optimality*, volume 49, pages 210–228. Institute of Mathematical Statistics Lecture Notes-Monograph Series. arxiv.org/abs/math.ST/0611192.

Loh, W.-Y. (2008a). Classification and regression tree methods. In Ruggeri, F., Kenett, R., and Faltin, F. W., editors, *Encyclopedia of Statistics in Quality and Reliability*, pages 315–323. Wiley, Chichester, UK. www.stat.wisc.edu/~loh/treeprogs/guide/eqr.pdf.

Loh, W.-Y. (2008b). Regression by parts: Fitting visually interpretable models with GUIDE. In Chen, C., Härdle, W., and Unwin, A., editors, *Handbook of Computational Statistics*, pages 447–469. Springer. www.stat.wisc.edu/~loh/treeprogs/guide/handbk.pdf.

Loh, W.-Y. (2009). Improving the precision of classification trees. *Annals of Applied Statistics*, 3:1710–1737. www.stat.wisc.edu/~loh/treeprogs/guide/aoas260.pdf.

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1:14–23.

Loh, W.-Y. (2012). Variable selection for classification and regression in large p, small n problems. In Barbour, A., Chan, H. P., and Siegmund, D., editors, *Probability Approximations and Beyond*, volume 205 of *Lecture Notes in Statistics—Proceedings*, pages 133–157, New York. Springer.

Loh, W.-Y. (2014). Fifty years of classification and regression trees (with discussion). *International Statistical Review*, 34:329–370.

Loh, W.-Y., Chen, C.-W., and Zheng, W. (2007). Extrapolation errors in linear model trees. *ACM Trans. Knowl. Discov. Data*, 1(2):6. www.stat.wisc.edu/~loh/treeprogs/guide/acm.pdf.

Loh, W.-Y., He, X., and Man, M. (2015). A regression tree approach to identifying subgroups with differential treatment effects. *Statistics in Medicine*. DOI: 10.1002/sim.6454.

Loh, W.-Y. and Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica*, 7:815–840. www3.stat.sinica.edu.tw/statistica/j7n4/j7n41/j7n41.htm.

Loh, W.-Y. and Zheng, W. (2013). Regression trees for longitudinal and multiresponse data. *Annals of Applied Statistics*, 7:495–522.

Marc, L. G., Raue, P. J., and Bruce, M. L. (2008). Screening performance of the 15-item Geriatric Depression Scale in a diverse elderly home care population. *American Journal of Geriatric Psychiatry*, 16:914–921.

Murnane, R. J., Boudett, K. P., and Willett, J. B. (1999). Do male dropouts benefit from obtaining a GED, postsecondary education, and training? *Evaluation Reviews*, 23:475–502.

Peters, A. and Hothorn, T. (2012). Improved predictors. R package version 0.8-13.

Quinlan, J. R. (1992). Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rousseeuw, P. J. and Leroy, A. M. (1987). *Robust Regression and Outlier Detection*. Wiley.

Schmoor, C., Olschewski, M., and Schumacher, M. (1996). Randomized and non-randomized patients in clinical trials: experiences with comprehensive cohort studies. *Statistics in Medicine*, 15:263–271.

Singer, J. D. and Willett, J. B. (2003). *Applied Longitudinal Data Analysis*. Oxford University Press, New York, NY.

Witten, I. and Frank, E. (2000). *Data Mining: Practical Machine Learning Tools and Techniques with JAVA Implementations*. Morgan Kaufmann, San Fransico, CA. www.cs.waikato.ac.nz/ml/weka.

Yeh, I.-C. (2007). Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement and Concrete Composites*, 29:474–480.