Package 'tree'

February 20, 2015

Title Classification and Regression Trees

Version 1.0-35 **Date** 2015-02-06

Depends R (>= 2.15.0), grDevices, graphics, stats
Suggests MASS
Description Classification and regression trees.
LazyLoad yes
License GPL-2 GPL-3
NeedsCompilation yes
Author Brian Ripley [aut, cre]
Maintainer Brian Ripley <ripley@stats.ox.ac.uk></ripley@stats.ox.ac.uk>
Repository CRAN
Date/Publication 2015-02-06 22:31:16
R topics documented:
cv.tree
deviance.tree
misclass.tree
na.tree.replace
partition.tree
plot.tree
plot.tree.sequence
predict.tree
snip.tree
text.tree
tile.tree
tree
tree.control
tree.screens
Index 19

2 cv.tree

cv.tree

Cross-validation for Choosing Tree Complexity

Description

Runs a K-fold cross-validation experiment to find the deviance or number of misclassifications as a function of the cost-complexity parameter k.

Usage

```
cv.tree(object, rand, FUN = prune.tree, K = 10, ...)
```

Arguments

object	An object of class "tree".
rand	Optionally an integer vector of the length the number of cases used to create object, assigning the cases to different groups for cross-validation.
FUN	The function to do the pruning.
K	The number of folds of the cross-validation.
	Additional arguments to FUN.

Value

A copy of FUN applied to object, with component dev replaced by the cross-validated results from the sum of the dev components of each fit.

Author(s)

```
B. D. Ripley
```

See Also

```
tree, prune.tree
```

Examples

deviance.tree 3

devi	ance	t	ree

Extract Deviance from a Tree Object

Description

Extract deviance from a tree object.

Usage

```
## S3 method for class 'tree'
deviance(object, detail = FALSE, ...)
```

Arguments

object an object of calls "tree"

detail logical. If true, returns a vector of deviance contributions from each node.

... arguments to be passed to or from other methods.

Value

The overall deviance, or a vector of contributions from the cases at each node. The overall deviance is the sum over leaves in the latter case.

misclass.tree

Misclassifications by a Classification Tree

Description

Report the number of mis-classifications made by a classification tree, either overall or at each node.

Usage

```
misclass.tree(tree, detail = FALSE)
```

Arguments

tree Object of class "tree", representing a classification tree.

detail If false, report overall number of mis-classifications. If true, report the number

at each node.

Details

The quantities returned are weighted by the observational weights if these are supplied in the construction of tree.

na.tree.replace

Value

Either the overall number of misclassifications or the number for each node.

Author(s)

```
B. D. Ripley
```

See Also

tree

Examples

```
ir.tr <- tree(Species ~., iris)
misclass.tree(ir.tr)
misclass.tree(ir.tr, detail=TRUE)</pre>
```

na.tree.replace

Replace NAs in Predictor Variables

Description

Adds a new level called "NA" to any discrete predictor in a data frame that contains NAs. Stops if any continuous predictor contains an NA.

Usage

```
na.tree.replace(frame)
```

Arguments

frame

data frame used to grow a tree.

Details

This function is used via the na.action argument to tree.

Value

data frame such that a new level named "NA" is added to any discrete predictor in frame with NAs.

See Also

```
tree, na.omit.
```

partition.tree 5

partition.tree	Plot the Partitions of a simple Tree Model	

Description

Plot the partitions of a tree involving one or two variables.

Usage

```
partition.tree(tree, label = "yval", add = FALSE, ordvars, ...)
```

Arguments

tree	A object of class "tree".
label	A character string giving the column of the frame component of tree to be used to label the regions.
add	If true, add to existing plot, otherwise start a new plot.
ordvars	The ordering of the variables to be used in a 2D plot. Specify the names in a character string of length 2; the first will be used on the x axis.
	Graphical parameters.

Details

This can be used with a regression or classification tree containing one or two continuous predictors (only).

If the tree contains one predictor, the predicted value (a regression tree) or the probability of the first class (a classification tree) is plotted against the predictor over its range in the training set.

If the tree contains two predictors, a plot is made of the space covered by those two predictors and the partition made by the tree is superimposed.

Value

None.

Author(s)

B. D. Ripley

See Also

tree

6 plot.tree

Examples

plot.tree

Plot a Tree Object

Description

Plot a tree object on the current graphical device

Usage

```
## S3 method for class 'tree'
plot(x, y = NULL, type = c("proportional", "uniform"), ...)
```

Arguments

x an object of class "tree".

y ignored. Used for positional matching of type.

type character string. If this partially matches "uniform", the branches are of uniform that the description in imposition

form length. Otherwise they are proportional to the decrease in impurity.

... graphical parameters.

Value

An (invisible) list with components x and y giving the coordinates of the tree nodes.

As a side effect, the value of type == "uniform" is stored in the variable .Tree.unif.? in the global environment, where ? is the device number.

Author(s)

B. D. Ripley

See Also

tree

plot.tree.sequence 7

plot.tree.sequence Plot a Tree Sequence

Description

Allows the user to plot a tree sequence.

Usage

Arguments

```
object of class tree.sequence. This is assumed to be the result of some func-
tion that produces an object with the same named components (size, deviance,
k) as that returned by prune.tree.

order of size on the plot. Use "decreasing" for the natural ordering of k and the
amount of pruning. Only the first character is needed.

type, ylim, ...
graphical parameters.
```

Details

This function is a method for the generic function plot() for class tree.sequence. It can be invoked by calling plot(x) for an object x of the appropriate class, or directly by calling plot. tree.sequence(x) regardless of the class of the object.

Side Effects

Plots deviance or number of misclassifications (or total loss) versus size for a sequence of trees.

Examples

8 predict.tree

		٠			
n	red	1	ct.	t.	ree

Predictions from a Fitted Tree Object

Description

Returns a vector of predicted responses from a fitted tree object.

Usage

Arguments

object	fitted model object of class tree. This is assumed to be the result of some function that produces an object with the same named components as that returned by the tree function.
newdata	data frame containing the values at which predictions are required. The predictors referred to in the right side of formula(object) must be present by name in newdata. If missing, fitted values are returned.
type	character string denoting whether the predictions are returned as a vector (default) or as a tree object.
split	governs the handling of missing values. If false, cases with missing values are dropped down the tree until a leaf is reached or a node for which the attribute is missing, and that node is used for prediction. If split = TRUE cases with missing attributes are split into fractional cases and dropped down each side of the split. The predicted values are averaged over the fractions to give the prediction.
nwts	weights for the newdata cases, used when predicting a tree.
eps	a lower bound for the probabilities, used if events of predicted probability zero occur in newdata when predicting a tree.
	further arguments passed to or from other methods.

Details

This function is a method for the generic function predict() for class tree. It can be invoked by calling predict(x) for an object x of the appropriate class, or directly by calling predict.tree(x) regardless of the class of the object.

prune.tree 9

Value

If type = "vector": vector of predicted responses or, if the response is a factor, matrix of predicted class probabilities. This new object is obtained by dropping newdata down object. For factor predictors, if an observation contains a level not used to grow the tree, it is left at the deepest possible node and frame\$yval or frame\$yprob at that node is the prediction.

If type = "tree": an object of class "tree" is returned with new values for frame\$n and frame\$dev. If newdata does not contain a column for the response in the formula the value of frame\$dev will be NA, and if some values in the response are missing, the some of the deviances will be NA.

If type = "class": for a classification tree, a factor of the predicted classes (that with highest posterior probability, with ties split randomly).

If type = "where": the nodes the cases reach.

References

Ripley, B. D. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge. Chapter 7.

See Also

```
predict, tree.
```

Examples

prune.tree

Cost-complexity Pruning of Tree Object

Description

Determines a nested sequence of subtrees of the supplied tree by recursively "snipping" off the least important splits.

Usage

10 prune.tree

Arguments

fitted model object of class tree. This is assumed to be the result of some functree tion that produces an object with the same named components as that returned by the tree() function. k cost-complexity parameter defining either a specific subtree of tree (k a scalar) or the (optional) sequence of subtrees minimizing the cost-complexity measure (k a vector). If missing, k is determined algorithmically. integer requesting the size (i.e. number of terminal nodes) of a specific subtree best in the cost-complexity sequence to be returned. This is an alternative way to select a subtree than by supplying a scalar cost-complexity parameter k. If there is no tree in the sequence of the requested size, the next largest is returned. data frame upon which the sequence of cost-complexity subtrees is evaluated. If newdata missing, the data used to grow the tree are used. weights for the newdata cases. nwts method character string denoting the measure of node heterogeneity used to guide costcomplexity pruning. For regression trees, only the default, deviance, is accepted. For classification trees, the default is deviance and the alternative is misclass (number of misclassifications or total loss). a matrix giving for each true class (row) the numeric loss of predicting the class loss (column). The classes should be in the order of the levels of the response. It is conventional for a loss matrix to have a zero diagonal. The default is 0–1 loss. eps a lower bound for the probabilities, used to compute deviances if events of predicted probability zero occur in newdata.

Details

Determines a nested sequence of subtrees of the supplied tree by recursively "snipping" off the least important splits, based upon the cost-complexity measure. prune.misclass is an abbreviation for prune.tree(method = "misclass") for use with cv.tree.

If k is supplied, the optimal subtree for that value is returned.

The response as well as the predictors referred to in the right side of the formula in tree must be present by name in newdata. These data are dropped down each tree in the cost-complexity sequence and deviances or losses calculated by comparing the supplied response to the prediction. The function cv.tree() routinely uses the newdata argument in cross-validating the pruning procedure. A plot method exists for objects of this class. It displays the value of the deviance, the number of misclassifications or the total loss for each subtree in the cost-complexity sequence. An additional axis displays the values of the cost-complexity parameter at each subtree.

Value

If k is supplied and is a scalar, a tree object is returned that minimizes the cost-complexity measure for that k. If best is supplied, a tree object of size best is returned. Otherwise, an object of class tree. sequence is returned. The object contains the following components:

size number of terminal nodes in each tree in the cost-complexity pruning sequence.

deviance total deviance of each tree in the cost-complexity pruning sequence.

k the value of the cost-complexity pruning parameter of each tree in the sequence.

snip.tree 11

Examples

```
data(fgl, package="MASS")
fgl.tr <- tree(type ~ ., fgl)
plot(print(fgl.tr))
fgl.cv <- cv.tree(fgl.tr,, prune.tree)
for(i in 2:5) fgl.cv$dev <- fgl.cv$dev +
    cv.tree(fgl.tr,, prune.tree)$dev
fgl.cv$dev <- fgl.cv$dev/5
plot(fgl.cv)</pre>
```

snip.tree

Snip Parts of Tree Objects

Description

snip. tree has two related functions. If nodes is supplied, it removes those nodes and all their descendants from the tree.

If nodes is not supplied, the user is invited to select nodes interactively; this makes sense only if the tree has already been plotted. A node is selected by clicking with the left mouse button; its number and the deviance of the current tree and that which would remain if that node were removed are printed. Selecting the same node again causes it to be removed (and the lines of its sub-tree erased). Clicking any other button terminates the selection process.

Usage

Arguments

tree	An object of class "tree".
nodes	An integer vector giving those nodes that are the roots of sub-trees to be snipped off. If missing, the user is invited to select a node at which to snip.
xy.save	If true, the \boldsymbol{x} and \boldsymbol{y} coordinates selected interactively are saved as attribute .xy of the returned value.
digits	Precision used in printing statistics for selected nodes.

Value

A tree object containing the nodes that remain after specified or selected subtrees have been snipped off.

Note

Prior to version 1.0-34, the saved coordinates were place in object .xy in the workspace.

text.tree

Author(s)

```
B. D. Ripley
```

See Also

```
tree, prune.tree.
```

text.tree

Annotate a Tree Plot

Description

Add text to a tree plot.

Usage

```
## S3 method for class 'tree'
text(x, splits = TRUE, label = "yval", all = FALSE,
    pretty = NULL, digits = getOption("digits") - 3,
    adj = par("adj"), xpd = TRUE, ...)
```

Arguments

X	an object of class "tree"
splits	logical. If TRUE the splits are labelled
label	The name of column in the frame component of x , to be used to label the nodes. Can be NULL to suppress node-labelling
all	logical. By default, only the leaves are labelled, but if true interior nodes are also labelled.
pretty	the manipulation used for split labels involving attributes. See Details.
digits	significant digits for numerical labels.
adj, xpd,	graphical parameters such as cex and font.

Details

If pretty = 0 then the level names of a factor split attributes are used unchanged. If pretty = NULL, the levels are presented by a, b, ...z, 0 ... 5. If pretty is a positive integer, abbreviate is applied to the labels with that value for its argument minlength.

If the lettering is vertical (par srt = 90) and adj is not supplied it is adjusted appropriately.

Value

None.

tile.tree

Author(s)

```
B. D. Ripley
```

See Also

```
plot.tree
```

Examples

```
ir.tr <- tree(Species ~., iris)
plot(ir.tr)
text(ir.tr)</pre>
```

tile.tree

Add Class Barcharts to a Classification Tree Plot

Description

This computes the frequencies of level of var for cases reaching each leaf of the tree, and plots barcharts of the set of frequencies underneath each leaf.

Usage

```
tile.tree(tree, var, screen.arg = ascr + 1, axes = TRUE)
```

Arguments

tree fitted object of class "tree".

var a factor variable to be displayed: by default it is the response factor of the tree.

screen.arg The screen to be used: default the next after the currently active screen.

axes logical flag for drawing of axes for the barcharts.

Value

A matrix of counts of categories (rows) for each leaf (columns). The principal effect is the plot.

Author(s)

B. D. Ripley

See Also

```
tree.screens
```

14 tree

Examples

```
data(fgl, package="MASS")
fgl.tr <- tree(type ~ ., fgl)
summary(fgl.tr)
plot(fgl.tr); text(fgl.tr, all=TRUE, cex=0.5)
fgl.tr1 <- snip.tree(fgl.tr, node=c(108, 31, 26))
tree.screens()
plot(fgl.tr1)
text(fgl.tr1)
tile.tree(fgl.tr1, fgl$type)
close.screen(all = TRUE)</pre>
```

tree

Fit a Classification or Regression Tree

Description

A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side.

Usage

```
tree(formula, data, weights, subset,
   na.action = na.pass, control = tree.control(nobs, ...),
   method = "recursive.partition",
   split = c("deviance", "gini"),
   model = FALSE, x = FALSE, y = TRUE, wts = TRUE, ...)
```

Arguments

formula	A formula expression. The left-hand-side (response) should be either a numerical vector when a regression tree will be fitted or a factor, when a classification tree is produced. The right-hand-side should be a series of numeric or factor variables separated by +; there should be no interaction terms. Both . and - are allowed: regression trees can have offset terms.
data	A data frame in which to preferentially interpret formula, weights and subset.
weights	Vector of non-negative observational weights; fractional weights are allowed.
subset	An expression specifying the subset of cases to be used.
na.action	A function to filter missing data from the model frame. The default is na.pass (to do nothing) as tree handles missing values (by dropping them down the tree as far as possible).
control	A list as returned by tree.control.
method	character string giving the method to use. The only other useful value is " $model.frame$ ".
split	Splitting criterion to use.

tree 15

model	If this argument is itself a model frame, then the formula and data arguments are ignored, and model is used to define the model. If the argument is logical and true, the model frame is stored as component model in the result.
Х	logical. If true, the matrix of variables for each case is returned.
У	logical. If true, the response variable is returned.
wts	logical. If true, the weights are returned.
	Additional arguments that are passed to tree.control. Normally used for mincut, minsize or mindev.

Details

A tree is grown by binary recursive partitioning using the response in the specified formula and choosing splits from the terms of the right-hand-side. Numeric variables are divided into X < a and X > a; the levels of an unordered factor are divided into two non-empty groups. The split which maximizes the reduction in impurity is chosen, the data set split and the process repeated. Splitting continues until the terminal nodes are too small or too few to be split.

Tree growth is limited to a depth of 31 by the use of integers to label nodes.

Factor predictor variables can have up to 32 levels. This limit is imposed for ease of labelling, but since their use in a classification tree with three or more levels in a response involves a search over $2^{(k-1)} - 1$ groupings for k levels, the practical limit is much less.

Value

The value is an object of class "tree" which has components

frame	A data frame with a row for each node, and row.names giving the node numbers. The columns include var, the variable used at the split (or " <leaf>" for a terminal node), n, the (weighted) number of cases reaching that node, dev the deviance of the node, yval, the fitted value at the node (the mean for regression trees, a majority class for classification trees) and split, a two-column matrix of the labels for the left and right splits at the node. Classification trees also have yprob, a matrix of fitted probabilities for each response level.</leaf>
where	An integer vector giving the row number of the frame detailing the node to which each case is assigned.
terms	The terms of the formula.
call	The matched call to Tree.
model	If model = TRUE, the model frame.
x	If $x = TRUE$, the model matrix.
У	If $y = TRUE$, the response.
wts	If wts = TRUE, the weights.

and attributes xlevels and, for classification trees, ylevels.

A tree with no splits is of class "singlenode" which inherits from class "tree".

16 tree.control

Author(s)

```
B. D. Ripley
```

References

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

Ripley, B. D. (1996) *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge. Chapter 7.

See Also

```
tree.control, prune.tree, predict.tree, snip.tree
```

Examples

```
data(cpus, package="MASS")
cpus.ltr <- tree(log10(perf) ~ syct+mmin+mmax+cach+chmin+chmax, cpus)
cpus.ltr
summary(cpus.ltr)
plot(cpus.ltr); text(cpus.ltr)
ir.tr <- tree(Species ~., iris)
ir.tr
summary(ir.tr)</pre>
```

tree.control

Select Parameters for Tree

Description

A utility function for use with the control argument of tree.

Usage

```
tree.control(nobs, mincut = 5, minsize = 10, mindev = 0.01)
```

Arguments

nobs	The number of observations in the training set.
mincut	The minimum number of observations to include in either child node. This is a weighted quantity; the observational weights are used to compute the 'number'. The default is 5.
minsize	The smallest allowed node size: a weighted quantity. The default is 10.
mindev	The within-node deviance must be at least this times that of the root node for the node to be split.

tree.screens 17

Details

This function produces default values of mincut and minsize, and ensures that mincut is at most half minsize.

To produce a tree that fits the data perfectly, set mindev = 0 and minsize = 2, if the limit on tree depth allows such a tree.

Value

A list:

mincut The maximum of the input or default mincut and 1 minsize The maximum of the input or default minsize and 2.

nmax A estimate of the maximum number of nodes that might be grown.

nobs The input nobs.

Note

The interpretation of mindev given here is that of Chambers and Hastie (1992, p. 415), and apparently not what is actually implemented in S. It seems S uses an absolute bound.

Author(s)

B. D. Ripley

See Also

tree

tree.screens Split Screen for Plotting Trees
--

Description

Splits the screen in a way suitable for using tile.tree.

Usage

```
tree.screens(figs, screen.arg = 0, ...)
```

Arguments

figs A specification of the split of the screen. See split.screen for the allowed

forms.

screen.arg the screen to divide, by default the whole display area.

... plot parameters to be passed to par.

18 tree.screens

Value

A vector of screen numbers for the newly-created screens.

Author(s)

B. D. Ripley

See Also

```
tile.tree, split.screen
```

Examples

```
data(fgl, package="MASS")
fgl.tr <- tree(type ~ ., fgl)
summary(fgl.tr)
plot(fgl.tr); text(fgl.tr, all=TRUE, cex=0.5)
fgl.tr1 <- snip.tree(fgl.tr, node=c(108, 31, 26))
tree.screens()
plot(fgl.tr1)
tile.tree(fgl.tr1, fgl$type)
close.screen(all = TRUE)</pre>
```

Index

```
*Topic hplot
                                                 print.tree (tree), 14
    partition.tree, 5
                                                 prune.misclass (prune.tree), 9
    plot.tree, 6
                                                 prune.tree, 2, 7, 9, 12, 16
    text.tree, 12
                                                 residuals.tree (tree), 14
    tile.tree, 13
    tree.screens, 17
                                                 snip.tree, 11, 16
*Topic tree
                                                 split.screen, 17, 18
    cv.tree, 2
                                                 summary.tree(tree), 14
    deviance.tree, 3
    misclass.tree, 3
                                                 text.tree, 12
    na.tree.replace, 4
                                                 tile.tree, 13, 18
    partition.tree, 5
                                                 tree, 2, 4-6, 9, 12, 14, 17
    plot.tree, 6
                                                 tree.control, 16, 16
    plot.tree.sequence, 7
                                                 tree.screens, 13, 17
    predict.tree, 8
    prune.tree, 9
    snip.tree, 11
    text.tree, 12
    tile.tree, 13
    tree, 14
    tree.control, 16
    tree.screens, 17
abbreviate, 12
cv.tree, 2, 10
deviance.singlenode (deviance.tree), 3
deviance.tree, 3
misclass.tree, 3
na.omit, 4
na.tree.replace, 4
partition.tree, 5
plot.tree, 6, 13
plot.tree.sequence, 7
predict, 9
predict.tree, 8, 16
print.summary.tree(tree), 14
```