

Package ‘mvpart’

April 17, 2009

Version 1.2-6

Date 2007-09-30

Author rpart by Terry M Therneau and Beth Atkinson <atkinson@mayo.edu>. R port of rpart by Brian Ripley <ripley@stats.ox.ac.uk>. Some routines from vegan – Jari Oksanen <jari.oksanen@oulu.fi> Extensions and adaptations of rpart to mvpart by Glenn De’ath <g.death@aims.gov.au>.

Maintainer Glenn De’ath <g.death@aims.gov.au>

Description Multivariate regression trees

Title Multivariate partitioning

Depends R (>= 2.2.0)

Suggests survival

License GPL-2 | file LICENSE

Repository CRAN

Date/Publication 2007-10-12 10:27:25

R topics documented:

car.test.frame	2
cmds.diss	3
gdist	4
kyphosis	6
labels.rpart	7
makeForm	8
meanvar.rpart	9
mvpart	10
na.rpart	12
path.rpart	12
plot.rpart	13
plotcp	15

post.rpart	16
predict.rpart	17
print.rpart	19
printcp	20
prune.rpart	21
residuals.rpart	22
rpart	23
rpart.control	25
rpart.matrix	26
rpart.object	27
rpart.pca	28
rpconvert	29
rsq.rpart	30
scaler	30
snip.rpart	32
solder	33
spider	34
summary.rpart	34
text.rpart	35
trclcomp	37
xdiss	38
xpred.rpart	39

Index	41
--------------	-----------

car.test.frame	<i>Automobile Data from 'Consumer Reports' 1990</i>
----------------	---

Description

The `car.test.frame` data frame has 60 rows and 8 columns, giving data on makes of cars taken from the April, 1990 issue of *Consumer Reports*.

Usage

```
data(car.test.frame)
```

Format

This data frame contains the following columns:

Price a numeric vector giving the list price in US dollars of a standard model

Country of origin, a factor with levels France Germany Japan Japan/USA Korea Mexico Sweden USA

Reliability a numeric vector coded 1 to 5.

Mileage fuel consumption miles per US gallon, as tested.

Type a factor with levels Compact Large Medium Small Sporty Van

Weight kerb weight in pounds.

Disp. the engine capacity (displacement) in litres.

HP the net horsepower of the vehicle.

Source

Consumer Reports, April, 1990, pp. 235–288 quoted in

John M. Chambers and Trevor J. Hastie eds. (1992) *Statistical Models in S*, Wadsworth and Brooks/Cole, Pacific Grove, CA 1992, pp. 46–47.

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
summary(z.auto)
```

 cmds.diss

Classical Scaling of Dissimilarity Measures

Description

The function first computes the dissimilarity matrix according to the specified method – see [gdist](#) or [xdiss](#). The dissimilarities are then scaled using classical scaling – see [cmdscale](#). The returned matrix can be input into [rpart](#) or [mvpart](#) for multivariate regression tree splitting.

Usage

```
cmds.diss(data, k = ncol(data), x.use = FALSE, zero.chk = TRUE,
          plt = FALSE, plot.subset = FALSE, plot.subn = 5, ...)
```

Arguments

data	Data matrix
k	Number of vectors to be returned
x.use	Use extended dissimilarity?
zero.chk	Check for zero row sums – if zero ignore these rows according to method
plt	Plot the relationship between the dissimilarities and the distances calculated from the scaled output vectors.
plot.subset	Plot a subset of the points – useful for large data sets.
plot.subn	Controls how many points are plotted when <code>plot.subset=TRUE</code> . The number of points plotted is $750 + N * \text{plot.subn}$ where $N = \text{number of rows in data}$.
...	arguments passed to either <code>xdiss</code> or <code>gdist</code>

Details

The function knows the same dissimilarity indices as `gdist`. Plotting the relationship between the dissimilarities and the distances calculated from the scaled output vectors is useful in assessing potential loss of information. If the loss is high then the results from partitioning directly from the dissimilarity matrix using distance-base partitioning (see `dist` in `rpart`), and those obtained from partitioning the output of `cmds.diss` using multivariate regression trees (see `mrt` in `rpart`) can be substantial.

Author(s)

Glenn De'ath

Examples

```
data(spider)
dist.vecs <- cmds.diss(spider)

# comparing splitting using "dist" and "mrt" methods
# for euclidean distance the answers are identical :
# first using "mrt" on the data directly
mvpart(data.matrix(spider[,1:12])~water+twigs+reft+herbs+moss+sand, spider, method="mrt", size=

# now using "dist" -- note we need the full distance matrix squared
mvpart(gdist(spider[,1:12],meth="euc",full=TRUE,sq=TRUE)~water+twigs+reft+herbs+moss+sand, sp

# finally using "mrt" from the scaled dissimilarities.
mvpart(cmds.diss(spider[,1:12],meth="euc")~water+twigs+reft+herbs+moss+sand, spider, method="m

# try with some other measure of dissimilarity eg extended bray-curtis -- the result will di
# between methods
```

gdist

Dissimilarity Measures

Description

The function computes useful dissimilarity indices which are known to have a good rank-order relation with gradient separation and are thus efficient in community ordination with multidimensional scaling.

Usage

```
gdist(x, method="bray", keepdiag=FALSE, full=FALSE, sq=FALSE)
```

Arguments

x	Data matrix
method	Dissimilarity index
keepdiag	Compute and keep diagonals
full	Return the square dissimilarity matrix
sq	Square the dissimilarities – useful for distance-based partitioning

Details

The function knows the following dissimilarity indices:

euclidean	$d_{jk} = \sqrt{\sum_i (x_{ij} - x_{ik})^2}$
manhattan	$d_{jk} = \sum_i x_{ij} - x_{ik} $
gower	$d_{jk} = \sum_i \frac{ x_{ij} - x_{ik} }{\max_i - \min_i}$
canberra	$d_{jk} = \frac{1}{N-Z} \sum_i \frac{ x_{ij} - x_{ik} }{x_{ij} + x_{ik}}$
bray	$d_{jk} = \frac{\sum_i x_{ij} - x_{ik} }{\sum_i (x_{ij} + x_{ik})}$
kulczynski	$d_{jk} = 1 - 0.5 \left(\frac{\sum_i \min(x_{ij}, x_{ik})}{\sum_i x_{ij}} + \frac{\sum_i \min(x_{ij}, x_{ik})}{\sum_i x_{ik}} \right)$
maximum	$d_{jk} = \max_i x_{ij} - x_{ik} $
binary	$d_{jk} = \sum_i x_{ij} > 0 - x_{ik} > 0 $
chord	$d_{jk} = \sqrt{\sum_i (x_{ij} - x_{ik})^2 / \sum_i (x_{ij} + x_{ik})^2}$

where $N - Z$ is the number of non-zero entries.

Infamous "double zeros" are removed in Canberra dissimilarity.

Euclidean and Manhattan dissimilarities are not good in gradient separation without proper standardization but are still included for comparison and special needs.

Some of indices become identical or rank-order similar after some standardizations.

Value

Should be interchangeable with `dist` and returns a distance object of the same type.

Note

The function is an alternative to `dist` adding some ecologically meaningful indices. Both methods should produce similar types of objects which can be interchanged in any method accepting either. Manhattan and Euclidean dissimilarities should be identical in both methods, and Canberra dissimilarity may be similar.

Author(s)

Jari Oksanen – modified Glenn De'ath (Dec 03)

References

Faith, D.P, Minchin, P.R. and Belbin, L. (1987) Compositional dissimilarity as a robust measure of ecological distance. *Vegetatio* 69, 57-68.

Examples

```
data(spider)
spider.dist <- gdist(spider[1:12,])
```

kyphosis

Data on Children who have had Corrective Spinal Surgery

Description

The `kyphosis` data frame has 81 rows and 4 columns. representing data on children who have had corrective spinal surgery

Usage

```
data(kyphosis)
```

Format

This data frame contains the following columns:

Kyphosis a factor with levels `absent` `present` indicating if a kyphosis (a type of deformation) was present after the operation.

Age in months

Number the number of vertebrae involved

Start the number of the first (topmost) vertebra operated on.

Source

John M. Chambers and Trevor J. Hastie eds. (1992) *Statistical Models in S*, Wadsworth and Brooks/Cole, Pacific Grove, CA 1992.

Examples

```
data(kyphosis)
fit <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis)
fit2 <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis,
             parms=list(prior=c(.65,.35), split='information'))
fit3 <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis,
             control=rpart.control(cp=.05))
par(mfrow=c(1,2))
plot(fit)
text(fit, use.n=TRUE)
plot(fit2)
text(fit2, use.n=TRUE)
```

Description

This function provides labels for the branches of an `rpart` tree.

Usage

```
## S3 method for class 'rpart':  
labels(object, digits=4, minlength=1, pretty, collapse=TRUE, ...)
```

Arguments

<code>object</code>	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
<code>digits</code>	the number of digits to be used for numeric values. All of the <code>rpart</code> functions that call <code>labels</code> explicitly set this value, with <code>options("digits")</code> as the default.
<code>minlength</code>	the minimum length for abbreviation of character or factor variables. If 0 no abbreviation is done; if 1 then single letters are used with "a" for the first level, "b" for the second and so on. If the value is greater than 1, the <code>abbreviate</code> function is used.
<code>pretty</code>	an argument included for backwards compatibility: <code>pretty=0</code> implies <code>minlength=0</code> , <code>pretty=NULL</code> implies <code>minlength=1</code> , and <code>pretty=TRUE</code> implies <code>minlength=4</code> .
<code>collapse</code>	logical. The returned set of labels is always of the same length as the number of nodes in the tree. If <code>collapse=TRUE</code> (default), the returned value is a vector of labels for the branch leading into each node, with "root" as the label for the top node. If <code>FALSE</code> , the returned value is a two column matrix of labels for the left and right branches leading out from each node, with "leaf" as the branch labels for terminal nodes.
<code>...</code>	optional arguments to <code>abbreviate</code> .

Value

Vector of split labels (`collapse=TRUE`) or matrix of left and right splits (`collapse=FALSE`) for the supplied `rpart` object. This function is called by printing methods for `rpart` and is not intended to be called directly by the users.

See Also

[abbreviate](#)

 makeForm

Formula Constructor

Description

This function constructs a formula from a data frame given the locations of the response and other variables. It is useful for long or repetitive formulas.

Usage

```
makeForm(data, ycol, xcol, zcol, FUN, maxy = 20, extra)
```

Arguments

<code>data</code>	The data frame or matrix for which the formula is to be constructed,
<code>ycol</code>	The locations (or names) of the response variables in <code>data</code> .
<code>xcol</code>	The locations (or names) of the explanatory variables in <code>data</code> .
<code>zcol</code>	The locations (or names) of the conditioning variables in <code>data</code> ; for use in the <code>vegan</code> package.
<code>FUN</code>	A function to apply to the response.
<code>maxy</code>	The number of multivariate responses before abbreviated notation is used. see examples below.
<code>extra</code>	Extra term(s) for the RHS of the formula.

Details

`makeForm` constructs a formula and is useful for long and/or repetitive formulae. See examples below.

Value

Returns a formula.

Examples

```
data(spider)
makeForm(spider,1,13:18)
# arct.lute ~ water + sand + moss + reft + twigs + herbs

makeForm(spider,1:12,13:18)
# cbind(arct.lute, pard.lugu, zora.spin, pard.nigr, pard.pull,
# aulo.albi, troc.terr, alop.cune, pard.mont, alop.acce, alop.fabr,
# arct.peri) ~ water + sand + moss + reft + twigs + herbs

makeForm(spider,1:12,13:15,16:18)
# cbind(arct.lute, pard.lugu, zora.spin, pard.nigr, pard.pull,
# aulo.albi, troc.terr, alop.cune, pard.mont, alop.acce, alop.fabr,
```

```

# arct.peri) ~ water + sand + moss + Condition(reft + twigs + herbs)

makeForm(spider, 1:12, 13:15, maxy=6)
# as.matrix(spider[, 1:12]) ~ water + sand + moss

makeForm(spider, 1:3, 13:15, FUN=sqrt)
# sqrt(cbind(arct.lute, pard.lugu, zora.spin)) ~ water + sand + moss

makeForm(spider, 1:3, 13:15, FUN=sqrt, extra="I(water^2)+")
# sqrt(cbind(arct.lute, pard.lugu, zora.spin)) ~ I(water^2) + water +
# sand + moss

```

meanvar.rpart

Mean-Variance Plot for an Rpart Object

Description

Creates a plot on the current graphics device of the deviance of the node divided by the number of observations at the node. Also returns the node number.

Usage

```

## S3 method for class 'rpart':
meanvar(tree, xlab="ave(y)", ylab="ave(deviance)", ...)

```

Arguments

tree	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
xlab	x-axis label for the plot.
ylab	y-axis label for the plot.
...	additional graphical parameters may be supplied as arguments to this function.

Value

an invisible list containing the following vectors is returned.

x	fitted value at terminal nodes (<code>yval</code>).
y	deviance of node divided by number of observations at node.
label	node number.

Side Effects

a plot is put on the current graphics device.

See Also

[plot.rpart.](#)

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
meanvar(z.auto, log='xy')
```

mvpart

Recursive Partitioning and Regression Trees

Description

Wrapper function for fitting and plotting rpart models

Usage

```
mvpart(form, data, minauto = TRUE, size, xv = c("lse", "min",
  "pick", "none"), xval = 10, xvmult = 0, xvse = 1, snip = FALSE,
  plot.add = TRUE, text.add = TRUE, digits = 3, margin = 0,
  uniform = FALSE, which = 4, pretty = TRUE, use.n = TRUE,
  all.leaves = FALSE, bars = TRUE, legend, bord = FALSE,
  xadj = 1, yadj = 1, prn = FALSE, branch = 1, rsq = FALSE,
  big.pts = FALSE, pca = FALSE, interact.pca = FALSE,
  wgt.ave.pca = FALSE, keep.y = TRUE, ...)
```

Arguments

form	As for rpart function. Arguments to rpart can be passed by
data	Optional data frame in which to interpret the variables named in the formula
minauto	If TRUE uses smart minsplit and minbucket based on N cases.
size	The size of tree to be generated.
xv	Selection of tree by cross-validation: "lse" - gives best tree within one SE of the overall best, "min" - the best tree, "pick" - pick the tree size interactively, "none" - no cross-validation.
xval	Number of cross-validations or vector defining cross-validation groups.
xvmult	Number of multiple cross-validations.
xvse	Multiplier for the number of SEs used for xv = "lse".
plot.add	Plot the tree and (optionally) add text.
text.add	Add output of text.rpart to tree.
snip	Interactively prune the tree.
digits	Number of digits on labels.

margin	Margin around plot, 0.1 gives an extra 10 percent space around the plot.
uniform	Uniform lengths to the branches of the tree.
which	Which split labels and where to plot them, 1=centered, 2 = left, 3 = right and 4 = both.
pretty	Pretty labels or full labels.
use.n	Add number of cases at each node.
all.leaves	Annotate all nodes.
bars	If TRUE adds barplots to nodes.
legend	If TRUE adds legend for mrt and classification trees.
bord	Border (box) around the barplots.
xadj, yadj	Adjust the size of the individual barplots (default = 1).
prn	If TRUE prints tree details.
branch	Controls spread of branches: 1=vertical lines, 0=maximum slope.
rsq	If TRUE gives "rsq" plot.
big.pts	Plot colored points at leaves – useful to link to PCA plot.
pca	If TRUE plots PCA of group means and add species and site information.
interact.pca	If TRUE runs interactive PCA. See <code>rpart.pca</code> .
wgt.ave.pca	If TRUE plot weighted averages across sites for species.
keep.y	If TRUE y values are returned.
...	... other arguments passed to <code>rpart</code> .

Value

an object of class `rpart`, a superset of class `tree`.

See Also

[rpart](#), [rpart.pca](#),

Examples

```
data(spider)
mvpart(data.matrix(spider[,1:12])~herbs+reft+moss+sand+twigs+water,spider) # defaults
mvpart(data.matrix(spider[,1:12])~herbs+reft+moss+sand+twigs+water,spider,xv="p") # pick th
# pick cv size and do PCA
fit <- mvpart(data.matrix(spider[,1:12])~herbs+reft+moss+sand+twigs+water,spider,xv="lse",pc
rpart.pca(fit,interact=TRUE,wgt.ave=TRUE) # interactive PCA plot of saved multivariate tree
```

<code>na.rpart</code>	<i>Handles Missing Values in an Rpart Object</i>
-----------------------	--

Description

Handles missing values in an `rpart` object.

Usage

```
na.rpart(x)
```

Arguments

`x` a model frame.

Details

Internal function that handles missing values when calling the function `rpart`.

<code>path.rpart</code>	<i>Follow Paths to Selected Nodes of an Rpart Object</i>
-------------------------	--

Description

Returns a names list where each element contains the splits on the path from the root to the selected nodes.

Usage

```
path.rpart(tree, nodes, pretty=0, print.it=TRUE)
```

Arguments

<code>tree</code>	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
<code>nodes</code>	an integer vector containing indices (node numbers) of all nodes for which paths are desired. If missing, user selects nodes as described below.
<code>pretty</code>	an integer denoting the extent to which factor levels in split labels will be abbreviated. A value of (0) signifies no abbreviation. A <code>NULL</code> , the default, signifies using elements of letters to represent the different factor levels.
<code>print.it</code>	Logical. Denotes whether paths will be printed out as nodes are interactively selected. Irrelevant if <code>nodes</code> argument is supplied.

Details

The function has a required argument as an `rpart` object and a list of nodes as optional arguments. Omitting a list of nodes will cause the function to wait for the user to select nodes from the dendrogram. It will return a list, with one component for each node specified or selected. The component contains the sequence of splits leading to that node. In the graphical interaction, the individual paths are printed out as nodes are selected.

Value

A named (by node) list, each element of which contains all the splits on the path from the root to the specified or selected nodes.

Graphical Interaction

A dendrogram of the `rpart` object is expected to be visible on the graphics device, and a graphics input device (eg a mouse) is required. Clicking (the selection button) on a node selects that node. This process may be repeated any number of times. Clicking the exit button will stop the selection process and return the list of paths.

References

This function was modified from `path.tree` in S.

See Also

[rpart](#)

Examples

```
data(kyphosis)
fit <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis)
summary(fit)
path.rpart(fit, node=c(11, 22))
```

`plot.rpart`*Plot an Rpart Object*

Description

Plots an `rpart` object on the current graphics device.

Usage

```
## S3 method for class 'rpart':
plot(x, uniform=FALSE, branch=1, compress=FALSE, nspace,
     margin=0, minbranch=.3, bar,...)
```

Arguments

<code>x</code>	a fitted object of class <code>rpart</code> , containing a classification, regression, or rate tree.
<code>uniform</code>	if <code>TRUE</code> , uniform vertical spacing of the nodes is used; this may be less cluttered when fitting a large plot onto a page. The default is to use a non-uniform spacing proportional to the error in the fit.
<code>branch</code>	controls the shape of the branches from parent to child node. Any number from 0 to 1 is allowed. A value of 1 gives square shouldered branches, a value of 0 give V shaped branches, with other values being intermediate.
<code>compress</code>	If <code>FALSE</code> , the leaf nodes will be at the horizontal plot coordinates of <code>1:nleaves</code> . If <code>TRUE</code> , the routine attempts a more compact arrangement of the tree. The compaction algorithm assumes <code>uniform=TRUE</code> ; surprisingly, the result is usually an improvement even when that is not the case.
<code>nospace</code>	the amount of extra space between a node with children and a leaf, as compared to the minimal space between leaves. Applies to compressed trees only. The default is the value of <code>branch</code> .
<code>margin</code>	an extra percentage of white space to leave around the borders of the tree. (Long labels sometimes get cut off by the default computation).
<code>minbranch</code>	set the minimum length for a branch to <code>minbranch</code> times the average branch length. This parameter is ignored if <code>uniform=TRUE</code> . Sometimes a split will give very little improvement, or even (in the classification case) no improvement at all. A tree with branch lengths strictly proportional to improvement leaves no room to squeeze in node labels.
<code>bar</code>	length of bar at root (default = 0.03) – used instead of char "I"
<code>...</code>	arguments to be passed to or from other methods.

Details

This function is a method for the generic function `plot`, for objects of class `rpart`. The y-coordinate of the top node of the tree will always be 1.

Value

the coordinates of the nodes are returned as a list, with components `x` and `y`.

Side Effects

an unlabeled plot is produced on the current graphics device.

See Also

[rpart](#), [text.rpart](#)

Examples

```
data(car.test.frame)
fit <- rpart(Price ~ Mileage + Type + Country, car.test.frame)
plot(fit, compress=TRUE)
text(fit, use.n=TRUE)
```

plotcp

*Plot a Complexity Parameter Table for an Rpart Fit***Description**

Gives a visual representation of the cross-validation results in an `rpart` object.

Usage

```
plotcp(x, xvse = 1, minline = TRUE, upper = c("size",
      "splits", "none"), tab, resub.err = TRUE, adj.df = FALSE, ...)
```

Arguments

<code>x</code>	an object of class <code>rpart</code>
<code>xvse</code>	multiplier for <code>xvse * SE</code> above the minimum of the curve.
<code>minline</code>	whether a horizontal line is drawn 1SE above the minimum of the curve.
<code>upper</code>	what is plotted on the top axis: the size of the tree (the number of leaves), the number of splits or nothing.
<code>tab</code>	used for multiple cross-validation.
<code>resub.err</code>	use resubstitution error for calculations of SEs.
<code>adj.df</code>	adjust df of resubstitution error estimate for calculations of SEs.
<code>...</code>	additional plotting parameters

Details

The set of possible cost-complexity prunings of a tree from a nested set. For the geometric means of the intervals of values of `cp` for which a pruning is optimal, a cross-validation has (usually) been done in the initial construction by `rpart`. The `cptable` in the fit contains the mean and standard deviation of the errors in the cross-validated prediction against each of the geometric means, and these are plotted by this function. A good choice of `cp` for pruning is often the leftmost value for which the mean lies below the horizontal line.

Value

None.

Side Effects

A plot is produced on the current graphical device.

See Also

[rpart](#), [printcp](#), [rpart.object](#)

post.rpart

PostScript Presentation Plot of an Rpart Object

Description

Generates a PostScript presentation plot of an `rpart` object.

Usage

```
## S3 method for class 'rpart':
post(tree, title.,
      filename = paste(deparse(substitute(tree)), ".ps", sep = ""),
      digits = getOption("digits") - 3, pretty = TRUE,
      use.n = TRUE, horizontal = TRUE, ...)
```

Arguments

<code>tree</code>	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
<code>title.</code>	a title which appears at the top of the plot. By default, the name of the <code>rpart</code> endpoint is printed out.
<code>filename</code>	ASCII file to contain the output. By default, the name of the file is the name of the object given by <code>rpart</code> (with the suffix <code>.ps</code> added). If <code>filename = ""</code> , the plot appears on the current graphical device.
<code>digits</code>	number of significant digits to include in numerical data.
<code>pretty</code>	an integer denoting the extent to which factor levels will be abbreviated in the character strings defining the splits; (0) signifies no abbreviation of levels. A <code>NULL</code> signifies using elements of letters to represent the different factor levels. The default (<code>TRUE</code>) indicates the maximum possible abbreviation.
<code>use.n</code>	Logical. If <code>TRUE</code> (default), adds to label (<code>#events level1/ #events level2/etc.</code> for method <code>class</code> , <code>n</code> for method <code>anova</code> , and <code>#events/n</code> for methods <code>poisson</code> and <code>exp</code>).
<code>horizontal</code>	Logical. If <code>TRUE</code> (default), plot is horizontal. If <code>FALSE</code> , plot appears as landscape.
<code>...</code>	other arguments to the <code>postscript</code> function.

Details

The plot created uses the functions `plot.rpart` and `text.rpart` (with the `fancy` option). The settings were chosen because they looked good to us, but other options may be better, depending on the `rpart` object. Users are encouraged to write their own function containing favorite options.

Side Effects

a plot of `rpart` is created using the `postscript` driver, or the current device if `filename = ""`.

See Also

[plot.rpart](#), [rpart](#), [text.rpart](#), [abbreviate](#)

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
post(z.auto, file = "") # display tree on active device
# now construct postscript version on file "pretty.ps"
# with no title
post(z.auto, file = "pretty.ps", title = " ")
z.hp <- rpart(Mileage ~ Weight + HP, car.test.frame)
post(z.hp)
```

predict.rpart

Predictions from a Fitted Rpart Object

Description

Returns a vector of predicted responses from a fitted `rpart` object.

Usage

```
## S3 method for class 'rpart':
predict(object, newdata=list(),
        type=c("vector", "prob", "class", "matrix", "where"), ...)
```

Arguments

<code>object</code>	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
<code>newdata</code>	data frame containing the values at which predictions are required. The predictors referred to in the right side of <code>formula(object)</code> must be present by name in <code>newdata</code> . If missing, the fitted values are returned.
<code>type</code>	character string denoting the type of predicted value returned. If the <code>rpart</code> object is a classification tree, then the default is to return <code>prob</code> predictions, a matrix whose columns are the probability of the first, second, etc. class. (This agrees with the default behavior of <code>tree</code>). Otherwise, a vector result is returned.
<code>...</code>	further arguments passed to or from other methods.

Details

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

Value

A new object is obtained by dropping `newdata` down the 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` at the node is the prediction.

If `type="vector"`:

vector of predicted responses. For regression trees this is the mean response at the node, for Poisson trees it is the estimated response rate, and for classification trees it is the predicted class.

If `type="prob"`:

(for a classification tree) a matrix of class probabilities.

If `type="matrix"`:

a matrix of the full responses (`frame$yval2` if this exists, otherwise `frame$yval`). For regression trees, this is the mean response, for Poisson trees it is the response rate and the number of events at that node in the fitted tree, and for classification trees it is the concatenation of the predicted class, the class counts at that node in the fitted tree, and the class probabilities.

If `type="class"`:

(for a classification tree) a factor of classifications based on the responses.

See Also

[predict, rpart.object](#)

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
predict(z.auto)

data(kyphosis)
fit <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis)
predict(fit, type="prob") # class probabilities (default)
predict(fit, type="vector") # level numbers
predict(fit, type="class") # factor
predict(fit, type="matrix") # level number, class frequencies, probabilities

data(iris)
sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
fit <- rpart(Species ~ ., data=iris, subset=sub)
fit
table(predict(fit, iris[-sub,], type="class"), iris[-sub, "Species"])
```

print.rpart *Print an Rpart Object*

Description

This function prints an `rpart` object. It is a method for the generic function `print` of class `rpart`.

Usage

```
## S3 method for class 'rpart':  
print(x, minlength=0, spaces=2, cp, digits= getOption("digits"), ...)
```

Arguments

<code>x</code>	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
<code>minlength</code>	Controls the abbreviation of labels: see labels.rpart .
<code>spaces</code>	the number of spaces to indent nodes of increasing depth.
<code>digits</code>	the number of digits of numbers to print.
<code>cp</code>	prune all nodes with a complexity less than <code>cp</code> from the printout. Ignored if unspecified.
<code>...</code>	arguments to be passed to or from other methods.

Details

This function is a method for the generic function `print` for class `"rpart"`. It can be invoked by calling `print` for an object of the appropriate class, or directly by calling `print.rpart` regardless of the class of the object.

Side Effects

A semi-graphical layout of the contents of `x$frame` is printed. Indentation is used to convey the tree topology. Information for each node includes the node number, split, size, deviance, and fitted value. For the `"class"` method, the class probabilities are also printed.

See Also

[print](#), [rpart.object](#), [summary.rpart](#), [printcp](#)

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
z.auto
## Not run: node), split, n, deviance, yval
      * denotes terminal node

1) root 60 1354.58300 24.58333
  2) Weight>=2567.5 45 361.20000 22.46667
    4) Weight>=3087.5 22 61.31818 20.40909 *
    5) Weight<3087.5 23 117.65220 24.43478
      10) Weight>=2747.5 15 60.40000 23.80000 *
      11) Weight<2747.5 8 39.87500 25.62500 *
    3) Weight<2567.5 15 186.93330 30.93333 *
## End(Not run)
```

printcp

Displays CP table for Fitted Rpart Object

Description

Displays the cp table for fitted rpart object.

Usage

```
printcp(x, digits=getOption("digits") - 2)
```

Arguments

x	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
digits	the number of digits of numbers to print.

Details

Prints a table of optimal prunings based on a complexity parameter.

See Also

[summary.rpart](#), [rpart.object](#)

Examples

```

data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
printcp(z.auto)
## Not run:
Regression tree:
rpart(formula = Mileage ~ Weight, data = car.test.frame)

Variables actually used in tree construction:
[1] Weight

Root node error: 1354.6/60 = 22.576

      CP nsplit rel error  xerror   xstd
1 0.595349      0  1.00000 1.03436 0.178526
2 0.134528      1  0.40465 0.60508 0.105217
3 0.012828      2  0.27012 0.45153 0.083330
4 0.010000      3  0.25729 0.44826 0.076998
## End(Not run)

```

prune.rpart

*Cost-complexity Pruning of an Rpart Object***Description**

Determines a nested sequence of subtrees of the supplied `rpart` object by recursively snipping off the least important splits, based on the complexity parameter (`cp`).

Usage

```

## S3 method for class 'rpart':
prune(tree, cp, ...)

```

Arguments

<code>tree</code>	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
<code>cp</code>	Complexity parameter to which the <code>rpart</code> object will be trimmed.
<code>...</code>	further arguments passed to or from other methods.

Value

A new `rpart` object that is trimmed to the value `cp`.

See Also

[rpart](#)

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
zp <- prune(z.auto, cp=0.1)
plot(zp) #plot smaller rpart object
```

residuals.rpart *Residuals From a Fitted Rpart Object*

Description

Method for residuals for an rpart object.

Usage

```
## S3 method for class 'rpart':
residuals(object, type = c("usual", "pearson", "deviance"), ...)
```

Arguments

object	fitted model object of class "rpart".
type	Indicates the type of residual desired. For regression or anova trees all three residual definitions reduce to $y - \text{fitted}$. This is the residual returned for user method trees as well. For classification trees the usual residuals are the missclassification losses $L(\text{actual}, \text{predicted})$ where L is the loss matrix. With default losses this residual is 0/1 for correct/incorrect classification. The pearson residual is $(1 - \text{fitted})/\sqrt{\text{fitted}(1 - \text{fitted})}$ and the deviance residual is $\sqrt{\text{minus twice log-arithm of fitted}}$. For poisson and exp (or survival) trees, the usual residual is the observed - expected number of events. The pearson and deviance residuals are as defined in McCullagh and Nelder.
...	further arguments passed to or from other methods.

Value

vector of residuals of type type from a fitted rpart object.

References

McCullagh P. and Nelder, J. A. (1989) *Generalized Linear Models*. London: Chapman and Hall.

Examples

```
data(solder)
fit <- rpart(skips ~ Opening + Solder + Mask + PadType + Panel,
             data=solder, method='anova')
summary(residuals(fit))
plot(predict(fit), residuals(fit))
```

rpart

*Recursive Partitioning and Regression Trees***Description**

Fit a rpart model

Usage

```
rpart(formula, data=NULL, weights, subset, na.action=na.rpart, method,
      dissim, model=FALSE, x=FALSE, y=TRUE, parms, control, cost, ...)
```

Arguments

formula	a formula, as in the <code>lm</code> function.
data	an optional data frame in which to interpret the variables named in the formula
weights	optional case weights.
subset	optional expression saying that only a subset of the rows of the data should be used in the fit.
na.action	The default action deletes all observations for which <code>y</code> is missing, but keeps those in which one or more predictors are missing.
method	one of "anova", "poisson", "class", "mrt", "dist", or "exp". If <code>method</code> is missing then the routine tries to make an intelligent guess. If <code>y</code> is a survival object, then <code>method="exp"</code> is assumed, if <code>y</code> is a matrix then <code>method="mrt"</code> is assumed, if <code>y</code> is a factor then <code>method="class"</code> is assumed, otherwise <code>method="anova"</code> is assumed. It is wisest to specify the method directly, especially as more criteria are added to the function. For <code>method="dist"</code> the response must be a square symmetric distance matrix; e.g. returned by <code>gdiss</code> or <code>xdiss</code> . Weights and cross-validation are currently not implemented for <code>method="dist"</code> . Alternatively, <code>method</code> can be a list of functions named <code>init</code> , <code>split</code> and <code>eval</code> .
dissim	used when <code>method="anova"</code> or <code>method="mrt"</code> . Dissimilarity types are either "euc" for Euclidean (sums of squares about the mean) or "man" for Manhattan (sums of absolute deviations about the mean). The latter is experimental and has proved useful for ecological data.

<code>model</code>	keep a copy of the model frame in the result. If the input value for <code>model</code> is a model frame (likely from an earlier call to the <code>rpart</code> function), then this frame is used rather than constructing new data.
<code>x</code>	keep a copy of the <code>x</code> matrix in the result.
<code>y</code>	keep a copy of the dependent variable in the result.
<code>parms</code>	optional parameters for the splitting function. Anova splitting has no parameters. Poisson splitting has a single parameter, the coefficient of variation of the prior distribution on the rates. The default value is 1. Exponential splitting has the same parameter as Poisson. For classification splitting, the list can contain any of: the vector of prior probabilities (component <code>prior</code>), the loss matrix (component <code>loss</code>) or the splitting index (component <code>split</code>). The priors must be positive and sum to 1. The loss matrix must have zeros on the diagonal and positive off-diagonal elements. The splitting index can be <code>gini</code> or <code>information</code> . The default priors are proportional to the data counts, the losses default to 1, and the split defaults to <code>gini</code> .
<code>control</code>	options that control details of the <code>rpart</code> algorithm.
<code>cost</code>	a vector of non-negative costs, one for each variable in the model. Defaults to one for all variables. These are scalings to be applied when considering splits, so the improvement on splitting on a variable is divided by its cost in deciding which split to choose.
<code>...</code>	arguments to <code>rpart.control</code> may also be specified in the call to <code>rpart</code> . They are checked against the list of valid arguments.

Details

This differs from the `tree` function mainly in its handling of surrogate variables. In most details it follows Breiman et. al. quite closely.

Value

an object of class `rpart`, a superset of class `tree`.

References

- Breiman, Friedman, Olshen, and Stone. (1984) *Classification and Regression Trees*. Wadsworth.
- De'ath G. (2002) Multivariate Regression Trees : A New Technique for Constrained Classification Analysis. *Ecology* 83(4):1103-1117.

See Also

[rpart.control](#), [rpart.object](#), [summary.rpart](#), [print.rpart](#)

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
summary(z.auto)
```

```

plot(z.auto); text(z.auto)

data(spider)
fit1 <- rpart(data.matrix(spider[,1:12])~water+twigs+reft+herbs+moss+sand, spider, method="mrt")
plot(fit1); text(fit1)
fit2 <- rpart(data.matrix(spider[,1:12])~water+twigs+reft+herbs+moss+sand, spider, method="mrt")
plot(fit2); text(fit2)
fit3 <- rpart(gdist(spider[,1:12],meth="bray",full=TRUE,sq=TRUE)~water+twigs+reft+herbs+moss+sand, spider, method="mrt")
plot(fit3); text(fit3)

```

rpart.control

Control for Rpart Models

Description

Various parameters that control aspects of the `rpart` fit.

Usage

```

rpart.control(minsplit=5, minbucket=round(minsplit/3), cp=0.01,
              maxcompete=4, maxsurrogate=0, usesurrogate=2, xval=10,
              surrogatestyle=0, maxdepth=30, ...)

```

Arguments

<code>minsplit</code>	the minimum number of observations that must exist in a node, in order for a split to be attempted.
<code>minbucket</code>	the minimum number of observations in any terminal <leaf> node. If only one of <code>minbucket</code> or <code>minsplit</code> is specified, the code either sets <code>minsplit</code> to <code>minbucket*3</code> or <code>minbucket</code> to <code>minsplit/3</code> , as appropriate.
<code>cp</code>	complexity parameter. Any split that does not decrease the overall lack of fit by a factor of <code>cp</code> is not attempted. For instance, with <code>anova</code> splitting, this means that the overall <code>Rsquare</code> must increase by <code>cp</code> at each step. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. Essentially, the user informs the program that any split which does not improve the fit by <code>cp</code> will likely be pruned off by cross-validation, and that hence the program need not pursue it.
<code>maxcompete</code>	the number of competitor splits retained in the output. It is useful to know not just which split was chosen, but which variable came in second, third, etc.
<code>maxsurrogate</code>	the number of surrogate splits retained in the output. If this is set to zero the compute time will be shortened, since approximately half of the computational time (other than setup) is used in the search for surrogate splits.
<code>usesurrogate</code>	how to use surrogates in the splitting process. 0= display only; an observation with a missing value for the primary split rule is not sent further down the tree. 1= use surrogates, in order, to split subjects missing the primary variable; if all surrogates are missing the observation is not split. 2= if all surrogates are missing, then send the observation in the majority direction. A value of 0 corresponds to the action of <code>tree</code> , and 2 to the recommendations of Breiman, et.al.

xval	number of cross-validations
surrogatestyle	controls the selection of a best surrogate. If set to 0 (default) the program uses the total number of correct classification for a potential surrogate variable, if set to 1 it uses the percent correct, calculated over the non-missing values of the surrogate. The first option more severely penalizes covariates with a large number of missing values.
maxdepth	Set the maximum depth of any node of the final tree, with the root node counted as depth 0 (past 30 rpart will give nonsense results on 32-bit machines).
...	mop up other arguments.

Value

a list containing the options.

See Also

[rpart](#)

rpart.matrix	<i>Creates a Model Matrix from a Call to Rpart</i>
--------------	--

Description

Creates a model matrix from a call to rpart.

Usage

```
rpart.matrix(frame)
```

Arguments

frame model frame (from call to rpart)

Value

Returns a matrix from the frame of class matrix. Used internally by the function rpart.

rpart.object *Recursive Partitioning and Regression Trees Object*

Description

These are objects representing fitted `rpart` trees.

Value

<code>frame</code>	<p>data frame with one row for each node in the tree. The <code>row.names</code> of <code>frame</code> contain the (unique) node numbers that follow a binary ordering indexed by node depth. Elements of <code>frame</code> include <code>var</code>, the variable used in the split at each node (leaf nodes are denoted by the string <code><leaf></code>), <code>n</code>, the size of each node, <code>wt</code>, the sum of case weights for the node, <code>dev</code>, the deviance of each node, <code>yval</code>, the fitted value of the response at each node, and <code>splits</code>, a two column matrix of left and right split labels for each node. All of these are the same as for an <code>rpart</code> object.</p> <p>Extra response information is in <code>yval2</code>, which contains the number of events at the node (poisson), or a matrix containing the fitted class, the class counts for each node and the class probabilities (classification). Also included in the frame are <code>complexity</code>, the complexity parameter at which this split will collapse, <code>ncompete</code>, the number of competitor splits retained, and <code>nsurrogate</code>, the number of surrogate splits retained.</p>
<code>where</code>	<p>vector, the same length as the number of observations in the root node, containing the row number of <code>frame</code> corresponding to the leaf node that each observation falls into.</p>
<code>splits</code>	<p>a matrix describing the splits. The row label is the name of the split variable, and columns are <code>count</code>, the number of observations sent left or right by the split (for competitor splits this is the number that would have been sent left or right had this split been used, for surrogate splits it is the number missing the primary split variable which were decided using this surrogate), <code>ncat</code>, the number of categories or levels for the variable ($+/-1$ for a continuous variable), <code>improve</code>, which is the improvement in deviance given by this split, or, for surrogates, the concordance of the surrogate with the primary, and <code>split</code>, the numeric split point. The last column <code>adj</code> gives the adjusted concordance for surrogate splits. For a factor, the <code>split</code> column contains the row number of the <code>csplit</code> matrix. For a continuous variable, the sign of <code>ncat</code> determines whether the subset $x < \text{cutpoint}$ or $x > \text{cutpoint}$ is sent to the left.</p>
<code>csplit</code>	<p>this will be present only if one of the split variables is a factor. There is one row for each such split, and column $i = -1$ if this level of the factor goes to the left, $+1$ if it goes to the right, and 0 if that level is not present at this node of the tree. For an ordered categorical variable all levels are marked as R/L, including levels that are not present.</p>
<code>method</code>	<p>the method used to grow the tree.</p>
<code>cptable</code>	<p>the table of optimal prunings based on a complexity parameter.</p>

terms	an object of mode <code>expression</code> and class <code>term</code> summarizing the formula. Used by various methods, but typically not of direct relevance to users.
call	an image of the call that produced the object, but with the arguments all named and with the actual formula included as the formula argument. To re-evaluate the call, say <code>update(tree)</code> . Optional components include the matrix of predictors (<code>x</code>) and the response variable (<code>y</code>) used to construct the <code>rpart</code> object.

Structure

The following components must be included in a legitimate `rpart` object. Of these, only the `where` component has the same length as the data used to fit the `rpart` object.

See Also

[rpart.](#)

`rpart.pca`

Principle Components Plot of a Multivariate Rpart Object

Description

Plots a PCA of the `rpart` object on the current graphics device.

Usage

```
## S3 method for class 'pca':
rpart(tree, pts = TRUE, plt.allx = TRUE, speclab = TRUE,
      specvecs = TRUE, wgt.ave = FALSE, add.tree = TRUE,
      cv1 = 1, cv2 = 2, chulls = TRUE, interact = FALSE, ...)
```

Arguments

<code>tree</code>	A fitted object of class <code>rpart</code> containing a multivariate regression tree.
<code>pts</code>	If <code>TRUE</code> , large points representing the leaf means are plotted.
<code>plt.allx</code>	If <code>TRUE</code> , small points representing individual cases are plotted.
<code>speclab</code>	If <code>TRUE</code> the labels of the response variables are plotted.
<code>specvecs</code>	If <code>TRUE</code> the vectors of the response variables are plotted provided <code>wgt.ave</code> is <code>FALSE</code> .
<code>wgt.ave</code>	If <code>TRUE</code> use weighted averages of responses not vectors.
<code>add.tree</code>	If <code>TRUE</code> add the tree structure to the plot.
<code>cv1</code>	Defines the principal component to plot horizontally – but see <code>interact</code> .
<code>cv2</code>	Defines the principal component to plot vertically – but see <code>interact</code> .
<code>chulls</code>	If <code>TRUE</code> adds convex hulls to the tree groups.
<code>interact</code>	If <code>TRUE</code> the plot can be viewed in dimensions by left-clicking to top-left, bottom-right or bottom-left (reset).
<code>...</code>	arguments to be passed to or from other methods.

Details

This function plots a PCA biplot of the group means (leaves) of multivariate regression objects of class `rpart`. The responses and group means and individual cases can be shown on the plot. If responses are positive (eg species-environment data) weighted averages of responses can be plotted.

Value

NULL

Side Effects

A PCA biplot plot is produced on the current graphics device.

See Also

`rpart`, `text.rpart`

Examples

```
data(spider)
fit<-mvpart(data.matrix(spider[,1:12])~herbs+reft+moss+sand+twigs+water,spider)
rpart.pca(fit)
rpart.pca(fit,wgt.ave=TRUE,interact=TRUE)
```

`rpconvert`

Update an rpart object

Description

Rpart objects changed (slightly) in their internal format in order to accomodate the changes for user-written split functions. This routine updates an old object to the new format.

Usage

```
rpconvert(x)
```

Arguments

`x` an `rpart` object

Value

an updated object

See Also

`rpart`

 rsq.rpart

Plots the Approximate R-Square for the Different Splits

Description

Produces 2 plots. The first plots the r-square (apparent and apparent - from cross-validation) versus the number of splits. The second plots the Relative Error(cross-validation) +/- 1-SE from cross-validation versus the number of splits.

Usage

```
rsq.rpart(x)
```

Arguments

`x` fitted model object of class `rpart`. This is assumed to be the result of some function that produces an object with the same named components as that returned by the `rpart` function.

Side Effects

Two plots are produced.

Note

The labels are only appropriate for the "anova" method.

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
rsq.rpart(z.auto)
```

 scaler

Row and Column Scaling of a Data Matrix

Description

The function provides some popular (and effective) standardization methods for community ecologists.

Usage

```
scaler(x, col = c("mean1", "max1", "min0", "ssq1", "range01", "zsc", "pa", "rank"),
      row = c("mean1", "max1", "min0", "ssq1", "range01", "zsc", "pa", "rank")[1])
```

Arguments

<code>x</code>	Data matrix.
<code>col</code>	Character vector of column standardizations.
<code>row</code>	Character vector of row standardizations.

Details

The function offers following data matrix standardizations:

- `mean1` : scale to mean of 1.
- `max1` : scale to maximum of 1.
- `ssq1` : scale to sums of squares equal 1.
- `range01` : scale range to 0-1.
- `zsc` : standardize to z-scores (mean=0, sd=1).
- `pa` : scale to presence/absence scale (0/1).
- `rank` : scale to rank order (1=lowest).

Standardizations are performed first on columns then on rows. "pa" applies to the whole matrix and can be specified using row or col.

Value

Returns the standardized matrix.

Note

Common transformations can be made with standard R functions.

Author(s)

Jari Oksanen – modified Glenn De'ath (Dec 03)

Examples

```
data(spider)
spid.data <- scaler(spider, col = "max", row="mean1")
```

`snip.rpart`*Snip Subtrees of an Rpart Object*

Description

Creates a "snipped" rpart object, containing the nodes that remain after selected subtrees have been snipped off. The user can snip nodes using the `toss` argument, or interactively by clicking the mouse button on specified nodes within the graphics window.

Usage

```
snip.rpart(x, toss)
```

Arguments

<code>x</code>	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
<code>toss</code>	an integer vector containing indices (node numbers) of all subtrees to be snipped off. If missing, user selects branches to snip off as described below.

Details

A dendrogram of `rpart` is expected to be visible on the graphics device, and a graphics input device (e.g., a mouse) is required. Clicking (the selection button) on a node displays the node number, sample size, response `yvalue`, and Error (`dev`). Clicking a second time on the same node snips that subtree off and visually erases the subtree. This process may be repeated an number of times. Warnings result from selecting the root or leaf nodes. Clicking the exit button will stop the snipping process and return the resulting `rpart` object.

See the documentation for the specific graphics device for details on graphical input techniques.

Value

a `rpart` object containing the nodes that remain after specified or selected subtrees have been snipped off.

Warning

Visually erasing the plot is done by over-plotting with the background colour. This will do nothing if the background is transparent (often true for screen devices).

See Also

[plot.rpart](#)

Examples

```
## dataset not in R
## Not run:
z.survey <- rpart(market.survey) #grow the rpart object
plot(z.survey) #plot the tree
z.survey2 <- snip.rpart(z.survey,toss=2) #trim subtree at node 2
plot(z.survey2) #plot new tree

# can also interactively select the node using the mouse in the
# graphics window
## End(Not run)
```

solder

Soldering of Components on Printed-Circuit Boards

Description

The solder data frame has 720 rows and 6 columns, representing a balanced subset of a designed experiment varying 5 factors on the soldering of components on printed-circuit boards.

Usage

```
data(solder)
```

Format

This data frame contains the following columns:

Opening a factor with levels L M S indicating the amount of clearance around the mounting pad.

Solder a factor with levels Thick Thin giving the thickness of the solder used.

Mask a factor with levels A1 .5 A3 B3 B6 indicating the type and thickness of mask used.

PadType a factor with levels D4 D6 D7 L4 L6 L7 L8 L9 W4 W9 giving the size and geometry of the mounting pad.

Panel 1 : 3 indicating the panel on a board being tested.

skips a numeric vector giving the number of visible solder skips.

Source

John M. Chambers and Trevor J. Hastie eds. (1992) *Statistical Models in S*, Wadsworth and Brooks/Cole, Pacific Grove, CA 1992.

Examples

```
data(solder)
fit <- rpart(skips ~ Opening + Solder + Mask + PadType + Panel,
            data=solder, method='anova')
summary(residuals(fit))
plot(predict(fit), residuals(fit))
```

spider *Spider Data*

Description

The `spider` data frame has 28 rows and 18 columns. The first 12 columns are abundances of different species of spiders and the next 6 are environmental data.

Usage

```
data(spider)
```

Format

This data frame contains the following columns:

arct.lute, **pard.lugu**, **zora.spin**, **pard.nigr**, **pard.pull**, **aulo.albi**, **troc.terr**, **alop.c**
 numeric vectors giving the abundances of 12 species of spider

water, **sand**, **moss**, **reft**, **twigs**, **herbs** numeric vectors giving the values of 6 environmental characteristics

Source

Van der Aart, P. J. and N. Smeek-Enserink. 1975. Correlations between distributions of hunting spiders (Lycosidae, Ctenidae) and environmental characteristics in a dune area. *Netherlands Journal of Zoology*. 25:1-45.

These data were analysed using multivariate trees in De'ath, G. 2002. Multivariate Regression Trees: A New Technique for Modelling Species-Environment Relationships. *Ecology*. 83(4):1103-1117

Examples

```
data(spider)
fit<-mvpart(as.matrix(spider[,1:12])~water+twigs+reft+herbs+moss+sand, spider)
summary(fit)
```

summary.rpart *Summarize a Fitted Rpart Object*

Description

Returns a detailed listing of a fitted `rpart` object.

Usage

```
## S3 method for class 'rpart':
summary(object, cp=0, digits=getOption("digits"), file, ...)
```

Arguments

object	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
digits	Number of significant digits to be used in the result.
cp	trim nodes with a complexity of less than <code>cp</code> from the listing.
file	write the output to a given file name. (Full listings of a tree are often quite long).
...	arguments to be passed to or from other methods.

Details

This function is a method for the generic function `summary` for class `"rpart"`. It can be invoked by calling `summary` for an object of the appropriate class, or directly by calling `summary.rpart` regardless of the class of the object.

See Also

`summary.rpart.object`, `printcp`.

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
summary(z.auto)
```

text.rpart

Place Text on a Dendrogram

Description

Labels the current plot of the tree dendrogram with text.

Usage

```
## S3 method for class 'rpart':
text(x, splits = TRUE, which = 4, label = "yval", FUN = text,
     all.leaves = FALSE, pretty = NULL, digits = getOption("digits") - 2,
     tadj = 0.65, stats = TRUE, use.n = FALSE, bars = TRUE,
     legend = FALSE, xadj = 1, yadj = 1, bord = FALSE, big.pts = FALSE, ...)
```

Arguments

<code>x</code>	fitted model object of class <code>rpart</code> . This is assumed to be the result of some function that produces an object with the same named components as that returned by the <code>rpart</code> function.
<code>splits</code>	logical flag. If <code>TRUE</code> (default), then the splits in the tree are labeled with the criterion for the split.
<code>which</code>	labels splits 1 = center, 2 = left, 3 = right, 4 = both.
<code>label</code>	a column name of <code>x\$frame</code> ; values of this will label the nodes. For the "class" method, <code>label="yval"</code> results in the factor levels being used, "yprob" results in the probability of the winning factor level being used, and 'specific yval level' results in the probability of that factor level.
<code>FUN</code>	the name of a labeling function, e.g. <code>text</code> .
<code>all.leaves</code>	Logical. If <code>TRUE</code> , all nodes are labeled, otherwise just terminal nodes.
<code>tadj</code>	Adjustment of text above (or below) splits.
<code>pretty</code>	an integer denoting the extent to which factor levels in split labels will be abbreviated. A value of (0) signifies no abbreviation. A <code>NULL</code> , the default, signifies using elements of letters to represent the different factor levels.
<code>digits</code>	number of significant digits to include in numerical labels.
<code>stats</code>	If <code>TRUE</code> adds statistics to nodes.
<code>use.n</code>	If <code>TRUE</code> adds <code>N</code> to labels. (<code>#events level1/ #events level2/etc.</code> for <code>class</code> , <code>n</code> for <code>anova</code> , and <code>#events/n</code> for <code>poisson</code> and <code>exp</code>).
<code>bars</code>	If <code>TRUE</code> adds barplots for multivariate regression trees.
<code>legend</code>	If <code>TRUE</code> adds legends for multivariate regression trees.
<code>xadj, yadj</code>	varies the size of barplots for multivariate regression trees.
<code>bord</code>	Adds borders (boxes) to barplots for multivariate regression trees.
<code>big.pts</code>	Adds color coded points to nodes. Useful to track groups to PCA plot (see <code>rpart.pca</code>).
<code>...</code>	Graphical parameters may also be supplied as arguments to this function (see <code>par</code>).

Side Effects

the current plot of a tree dendrogram is labeled.

See Also

[text](#), [plot.rpart](#), [rpart](#), [post.rpart](#), [abbreviate](#)

Examples

```
data(car.test.frame)
z.auto <- rpart(Mileage ~ Weight, car.test.frame)
plot(z.auto)
text(z.auto, use.n=TRUE, all=TRUE)
```

`trclcomp`*Tree-Clustering Comparison*

Description

This function compares the within-group variation for groups formed by tree partitioning and unconstrained clustering. The results are plotted and returned invisibly.

Usage

```
trclcomp(x, method = "com", km = TRUE, mrt = TRUE)
```

Arguments

<code>x</code>	Rpart object with method "mrt" – see rpart
<code>method</code>	The clustering method for the unconstrained clustering – see hclust
<code>km</code>	If TRUE a K-Means clustering is compared with the multivariate tree partitioning.
<code>mrt</code>	If TRUE an additional K-Means clustering with a starting configuration based on the multivariate tree partitioning is generated.

Details

The within-group variation for groups formed by multivariate tree partitioning and unconstrained clusterings are compared for all sizes of the hierarchy of tree partitions.

Value

Returns a list (invisibly) of the within-tree and within-cluster variation for all tree sizes.

Author(s)

Glenn De'ath

References

De'ath G. (2002) Multivariate Regression Trees : A New Technique for Constrained Classification Analysis. *Ecology* 83(4):1103-1117.

Examples

```
data(spider)
fit <- mvpart(data.matrix(spider[,1:12])~herbs+reft+moss+sand+twigs+water,spider)
trclcomp(fit)
```

`xdiss`*Extendend Dissimilarity Measures*

Description

The function computes extended dissimilarity indices which are for long gradients have better good rank-order relation with gradient separation and are thus efficient in community ordination with multidimensional scaling.

Usage

```
xdiss(data, dcrit = 1, dauto = TRUE, dinf = 0.5, method = "man",
      use.min = TRUE, eps = 1e-04, replace.neg = TRUE, big = 10000,
      sumry = TRUE, full = FALSE, sq = FALSE)
```

Arguments

<code>data</code>	Data matrix
<code>dcrit</code>	Dissimilarities < <code>dcrit</code> are considered to have no species in common and are recalculated.
<code>dauto</code>	Automatically select tuning parameters – recommended.
<code>method</code>	Dissimilarity index
<code>use.min</code>	Minimum dissimilarity of pairs of distances used – recommended.
<code>dinf</code> , <code>eps</code> , <code>replace.neg</code> , <code>big</code>	Internal parameters – leave as is usually.
<code>sumry</code>	Print summary of extended dissimilarities?
<code>full</code>	Return the square dissimilarity matrix.
<code>sq</code>	Square the dissimilarities – useful for distance-based partitionong.

Details

The function knows the same dissimilarity indices as [gdiss](#).

Value

Returns an object of class `distance` with attributes "Size" and "ok". "ok" is TRUE if rows are not disconnected (De'ath 1999).

Author(s)

Glenn De'ath

References

- De'ath, G. (1999) Extended dissimilarity: a method of robust estimation of ecological distances from high beta diversity data. *Plant Ecology* 144(2):191-199.
- Faith, D.P, Minchin, P.R. and Belbin, L. (1987) Compositional dissimilarity as a robust measure of ecological distance. *Vegetatio* 69, 57-68.

Examples

```
data(spider)
spider.dist <- xdiss(spider)
```

<code>xpred.rpart</code>	<i>Return Cross-Validated Predictions</i>
--------------------------	---

Description

Gives the predicted values for an `rpart` fit, under cross validation, for a set of complexity parameter values.

Usage

```
xpred.rpart(fit, xval=10, cp)
```

Arguments

<code>fit</code>	a <code>rpart</code> object.
<code>xval</code>	number of cross-validation groups. This may also be an explicit list of integers that define the cross-validation groups.
<code>cp</code>	the desired list of complexity values. By default it is taken from the <code>cp</code> table component of the fit.

Details

Complexity penalties are actually ranges, not values. If the `cp` values found in the table were .36, .28, and .13, for instance, this means that the first row of the table holds for all complexity penalties in the range $[.36, 1]$, the second row for `cp` in the range $[.28, .36)$ and the third row for $[.13, .28)$. By default, the geometric mean of each interval is used for cross validation.

Value

a matrix with one row for each observation and one column for each complexity value.

See Also

[rpart](#)

Examples

```
data(car.test.frame)
fit <- rpart(Mileage ~ Weight, car.test.frame)
xmat <- xpred.rpart(fit)
xerr <- (xmat - car.test.frame$Mileage)^2
apply(xerr, 2, sum) # cross-validated error estimate

# approx same result as rel. error from printcp(fit)
apply(xerr, 2, sum)/var(car.test.frame$Mileage)
printcp(fit)
```

Index

*Topic **datasets**

car.test.frame, 1
kyphosis, 5
solder, 32
spider, 33

*Topic **manip**

scaler, 29

*Topic **methods**

rpart.object, 26

*Topic **models**

makeForm, 7

*Topic **multivariate**

cmds.diss, 2
gdist, 4
scaler, 29
trclcomp, 36
xdiss, 37

*Topic **tree**

labels.rpart, 6
meanvar.rpart, 8
mvpart, 9
na.rpart, 11
path.rpart, 11
plot.rpart, 13
plotcp, 14
post.rpart, 15
predict.rpart, 16
print.rpart, 18
printcp, 19
prune.rpart, 20
residuals.rpart, 21
rpart, 22
rpart.control, 24
rpart.matrix, 25
rpart.object, 26
rpart.pca, 27
rpconvert, 28
rsq.rpart, 29
snip.rpart, 31

summary.rpart, 33
text.rpart, 34
xpred.rpart, 38

abbreviate, 7, 16, 35

car.test.frame, 1
cmds.diss, 2
cmdscale, 2

dist, 5

gdist, 2, 3, 4, 22, 37

hclust, 36

kyphosis, 5

labels.rpart, 6, 18

makeForm, 7
meanvar (*meanvar.rpart*), 8
meanvar.rpart, 8
mvpart, 2, 9

na.rpart, 11

path.rpart, 11
plot.rpart, 9, 13, 16, 31, 35
plotcp, 14
post (*post.rpart*), 15
post.rpart, 15, 35
pred.rpart (*predict.rpart*), 16
predict, 17
predict.rpart, 16
print, 19
print.rpart, 18, 23
printcp, 15, 19, 19, 34
prune (*prune.rpart*), 20
prune.rpart, 20

residuals.rpart, 21

`rpart`, 2, 3, 11, 12, 14–16, 21, 22, 25, 27, 28, 35, 36, 38
`rpart.control`, 23, 24
`rpart.matrix`, 25
`rpart.object`, 15, 17, 19, 20, 23, 26, 34
`rpart.pca`, 11, 27
`rpartcallback(rpart)`, 22
`rpconvert`, 28
`rsq.rpart`, 29

`scaler`, 29
`snip.rpart`, 31
`solder`, 32
`spider`, 33
`summary`, 34
`summary.rpart`, 19, 20, 23, 33

`text`, 35
`text.rpart`, 14, 16, 28, 34
`trclcomp`, 36
`tree`, 17

`xdiss`, 2, 22, 37
`xpred.rpart`, 38