

# Package ‘rpart’

August 5, 2009

**Priority** recommended

**Version** 3.1-45

**Date** 2009-07-28

**DateNote** March 2002 version of rpart

**Author** Terry M Therneau and Beth Atkinson <atkinson@mayo.edu>. R port by Brian Ripley.

**Maintainer** Brian Ripley <ripley@stats.ox.ac.uk>

**Description** Recursive partitioning and regression trees

**Title** Recursive Partitioning

**Depends** R (>= 2.7.0), graphics, stats, grDevices

**Suggests** survival

**License** GPL-2 | file LICENCE

**LazyLoad** yes

**LazyData** yes

**URL** <http://mayoresearch.mayo.edu/mayo/research/biostat/splusfunctions.cfm>

**Repository** CRAN

**Date/Publication** 2009-08-05 11:22:00

## R topics documented:

car.test.frame . . . . .	2
cu.summary . . . . .	3
kyphosis . . . . .	4
labels.rpart . . . . .	5
meanvar.rpart . . . . .	6
na.rpart . . . . .	7
path.rpart . . . . .	7

plot.rpart . . . . .	8
plotcp . . . . .	10
post.rpart . . . . .	11
predict.rpart . . . . .	12
print.rpart . . . . .	14
printcp . . . . .	15
prune.rpart . . . . .	16
residuals.rpart . . . . .	17
rpart . . . . .	18
rpart.control . . . . .	19
rpart.object . . . . .	21
rpconvert . . . . .	22
rsq.rpart . . . . .	23
snip.rpart . . . . .	23
solder . . . . .	25
summary.rpart . . . . .	26
text.rpart . . . . .	27
xpred.rpart . . . . .	28

<b>Index</b>	<b>30</b>
--------------	-----------

---

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*. This is part of a larger dataset, some columns of which are given in [cu.summary](#).

## Usage

```
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.

**See Also**

[cu.summary](#)

**Examples**

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

---

cu.summary

*Automobile Data from 'Consumer Reports' 1990*

---

**Description**

The `cu.summary` data frame has 117 rows and 5 columns, giving data on makes of cars taken from the April, 1990 issue of *Consumer Reports*.

**Usage**

```
cu.summary
```

**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 Brazil England France Germany Japan Japan/USA  
Korea Mexico Sweden USA

**Reliability** an ordered factor with levels Much worse < worse < average < better <  
Much better

**Mileage** fuel consumption miles per US gallon, as tested.

**Type** a factor with levels Compact Large Medium Small Sporty Van

**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.

**See Also**

[car.test.frame](#)

**Examples**

```
fit <- rpart(Price ~ Mileage + Type + Country, cu.summary)
plot(fit, compress=TRUE)
text(fit, use.n=TRUE)
```

---

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**

```
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**

```
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)
```

---

labels.rpart	<i>Create Split Labels For an Rpart Object</i>
--------------	--

---

**Description**

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

**Usage**

```
## S3 method for class 'rpart':  
labels(object, digits=4, minlength=1L, 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](#)

---

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

```
meanvar(tree, ...)  
  
## 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

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

---

`na.rpart`*Handles Missing Values in an Rpart Object*

---

**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`.

---

`path.rpart`*Follow Paths to Selected Nodes of an Rpart Object*

---

**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 (e.g. 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

```
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, ...)
```

**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>nspace</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>...</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**

```
fit <- rpart(Price ~ Mileage + Type + Country, cu.summary)
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, minline = TRUE, lty = 3, col = 1,
       upper = c("size", "splits", "none"), ...)
```

**Arguments**

<code>x</code>	an object of class <code>rpart</code>
<code>minline</code>	whether a horizontal line is drawn 1SE above the minimum of the curve.
<code>lty</code>	line type for this line
<code>col</code>	colour for this line
<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>...</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

```
post(tree, ...)

## 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**

```
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"),
        na.action = na.pass, ...)
```

**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>na.action</code>	a function to determine what should be done with missing values in <code>newdata</code> . The default is to pass them down the tree using surrogates in the way selected when the model was built. Other possibilities are <code>na.omit</code> and <code>na.fail</code> .
<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 (as a number).

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

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

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

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 <a href="#">labels.rpart</a> .
<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**

```

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**

```

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

```
prune(tree, ...)

## 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

```
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

<code>object</code>	fitted model object of class "rpart".
<code>type</code>	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 misclassification 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 $\sqrt{\text{minus twice logarithm of fitted}}$ . 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.
<code>...</code>	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

```
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, weights, subset, na.action = na.rpart, method,
      model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)
```

**Arguments**

<code>formula</code>	a formula, as in the <code>lm</code> function.
<code>data</code>	an optional data frame in which to interpret the variables named in the formula
<code>weights</code>	optional case weights.
<code>subset</code>	optional expression saying that only a subset of the rows of the data should be used in the fit.
<code>na.action</code>	The default action deletes all observations for which <code>y</code> is missing, but keeps those in which one or more predictors are missing.
<code>method</code>	one of "anova", "poisson", "class" 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> has 2 columns then <code>method="poisson"</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. Alternatively, <code>method</code> can be a list of functions named <code>init</code> , <code>split</code> and <code>eval</code> . Examples are given in the file 'tests/usersplits.R' in the sources.
<code>model</code>	if logical: 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. If missing and <code>model</code> is supplied this defaults to <code>FALSE</code> .
<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> .

control	options that control details of the <code>rpart</code> algorithm.
cost	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.
...	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.

### See Also

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

### Examples

```
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), xpd=NA) # otherwise on some devices the text is clipped
plot(fit)
text(fit, use.n=TRUE)
plot(fit2)
text(fit2, use.n=TRUE)
```

---

rpart.control

*Control for Rpart Models*

---

### Description

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

**Usage**

```
rpart.control(minsplit=20, minbucket=round(minsplit/3), cp=0.01,
              maxcompete=4, maxsurrogate=5, usesurrogate=2, xval=10,
              surrogatestyle=0, maxdepth=30, ...)
```

**Arguments**

minsplit	the minimum number of observations that must exist in a node, in order for a split to be attempted.
minbucket	the minimum number of observations in any terminal <leaf> node. If only one of minbucket or minsplit is specified, the code either sets minsplit to minbucket*3 or minbucket to minsplit/3, as appropriate.
cp	complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. For instance, with anova splitting, this means that the overall Rsquare must increase by cp 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 cp will likely be pruned off by cross-validation, and that hence the program need not pursue it.
maxcompete	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.
maxsurrogate	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.
usesurrogate	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 tree, 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](#)


---

<code>rpart.object</code>	<i>Recursive Partitioning and Regression Trees Object</i>
---------------------------	---

---

**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>, a factor giving the variable used in the split at each node (leaf nodes are denoted by the string <code>&lt;leaf&gt;</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>	integer 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.
<code>splits</code>	a numeric 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 ( $\pm 1$ 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.
<code>csplit</code>	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, $3$ if it goes to the right, and $2$ 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.

method	the method used to grow the tree.
cptable	the table of optimal prunings based on a complexity parameter.
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](#).

---

rpconvert

*Update an rpart object*

---

### Description

`Rpart` objects changed (slightly) in their internal format in order to accommodate 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**

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

---

`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  $y$ -value, 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

```
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

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

<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>	Number of significant digits to be used in the result.
<code>cp</code>	trim nodes with a complexity of less than <code>cp</code> from the listing.
<code>file</code>	write the output to a given file name. (Full listings of a tree are often quite long).
<code>...</code>	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

```
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, label, FUN=text, all=FALSE,
     pretty=NULL, digits=getOption("digits") - 3, use.n=FALSE,
     fancy=FALSE, fwidth=0.8, fheight=0.8, ...)
```

## 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.
splits	logical flag. If <code>TRUE</code> (default), then the splits in the tree are labeled with the criterion for the split.
label	For compatibility with <code>rpart2</code> , ignored in this version (with a warning).
FUN	the name of a labeling function, e.g. <code>text</code> .
all	Logical. If <code>TRUE</code> , all nodes are labeled, otherwise just terminal nodes.
pretty	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.
digits	number of significant digits to include in numerical labels.
use.n	Logical. If <code>TRUE</code> , adds to label ( <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> ).
fancy	Logical. If <code>TRUE</code> , nodes are represented by ellipses (interior nodes) and rectangles (leaves) and labeled by <code>yval</code> . The edges connecting the nodes are labeled by left and right splits.
fwidth	Relates to option <code>fancy</code> and the width of the ellipses and rectangles. If <code>fwidth &lt; 1</code> then it is a scaling factor (default = 0.8). If <code>fwidth &gt; 1</code> then it represents the number of character widths (for current graphical device) to use.
fheight	Relates to option <code>fancy</code> and the height of the ellipses and rectangles. If <code>fheight &lt; 1</code> then it is a scaling factor (default = 0.8). If <code>fheight &gt; 1</code> then it represents the number of character heights (for current graphical device) to use.
...	Graphical parameters may also be supplied as arguments to this function (see <code>par</code> ). As labels often extend outside the plot region it can be helpful to specify <code>xpd = TRUE</code> .

**Side Effects**

the current plot of a tree dendrogram is labeled.

**See Also**

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

**Examples**

```
freen.tr <- rpart(y ~ ., freeny)
plot(freen.tr)
text(freen.tr, use.n=TRUE, all=TRUE)
```

---

xpred.rpart

*Return Cross-Validated Predictions*

---

**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  $[\text{.36}, 1]$ , the second row for `cp` in the range  $[\text{.28}, \text{.36})$  and the third row for  $[\text{.13}, \text{.28})$ . 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**

```
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, 2  
cu.summary, 3  
kyphosis, 4  
solder, 25

## \*Topic **methods**

rpart.object, 21

## \*Topic **tree**

labels.rpart, 5  
meanvar.rpart, 6  
na.rpart, 7  
path.rpart, 7  
plot.rpart, 8  
plotcp, 10  
post.rpart, 11  
predict.rpart, 12  
print.rpart, 14  
printcp, 15  
prune.rpart, 16  
residuals.rpart, 17  
rpart, 18  
rpart.control, 19  
rpart.object, 21  
rpconvert, 22  
rsq.rpart, 23  
snip.rpart, 23  
summary.rpart, 26  
text.rpart, 27  
xpred.rpart, 28

abbreviate, 5, 12, 28

car.test.frame, 2, 3  
cu.summary, 2, 3

kyphosis, 4

labels.rpart, 5, 14

meanvar (*meanvar.rpart*), 6  
meanvar.rpart, 6

na.fail, 12

na.omit, 12

na.rpart, 7

path.rpart, 7

plot.rpart, 6, 8, 12, 24, 28

plotcp, 10

post (*post.rpart*), 11

post.rpart, 11, 28

predict, 13

predict.rpart, 12

print, 14

print.rpart, 14, 19

printcp, 10, 14, 15, 26

prune (*prune.rpart*), 16

prune.rpart, 16

residuals.rpart, 17

rpart, 8–10, 12, 16, 18, 21, 22, 28

rpart.control, 19, 19

rpart.object, 10, 13–15, 19, 21, 26

rpartcallback (*rpart*), 18

rpconvert, 22

rsq.rpart, 23

snip.rpart, 23

solder, 25

summary, 26

summary.rpart, 14, 15, 19, 26

text, 28

text.rpart, 9, 12, 27

tree, 12

xpred.rpart, 28