Package 'rpart.plot'

January 8, 2026

Version 3.1.4

Title Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'

Author Stephen Milborrow [aut, cre]

Maintainer Stephen Milborrow <milbo@sonic.net>

Depends R (>= 3.4.0), rpart (>= 4.1-15)

Suggests earth (>= 5.1.2)

Description Plot 'rpart' models. Extends plot.rpart() and text.rpart() in the 'rpart' package.

License GPL-3

LazyData yes

URL http://www.milbo.org/rpart-plot/index.html

NeedsCompilation no

Repository CRAN

Date/Publication 2026-01-08 08:40:53 UTC

Contents

prp 2
ptitanic 15
rpart.plot 17
rpart.predict 21
rpart.rules 22
show.prp.palettes 24

Index **25**

prp *Plot an rpart model.*

Description

Plot an rpart model.

First-time users should use rpart.plot instead, which provides a simplified interface to this function.

For an overview, please see the package vignette Plotting rpart trees with the rpart.plot package (also available here).

The arguments of this function are a superset of those of rpart.plot and some of the arguments have different defaults. In detail the different defaults are:

	rpart.plot	prp
type	2	0
extra	"auto"	0
fallen.leaves	TRUE	FALSE
varlen	0	-8
faclen	0	3
box.palette	"auto"	0

The defaults are different for historical reasons: for backwards compatibility the defaults of prp haven't changed, whereas the defaults of rpart.plot were changed when type="auto" and box.palette were introduced in version 2.0.0 of this package.

Usage

```
prp(x=stop("no 'x' arg"),
    type=0, extra=0, under=FALSE, fallen.leaves=FALSE,
    nn=FALSE, ni=FALSE, yesno=TRUE,
    branch=if(fallen.leaves) 1 else .2,
    uniform=TRUE, left=TRUE, xflip=FALSE, yflip=FALSE,
    digits=2, varlen=-8, faclen=3, roundint=TRUE,
    cex=NULL, tweak=1,
    clip.facs=FALSE, clip.right.labs=TRUE,
    compress=TRUE, ycompress=uniform,
    Margin=0, space=1, gap=NULL,
    snip=FALSE, snip.fun=NULL, trace=FALSE,

    box.col=0, box.palette=0,
    pal.thresh=NULL, pal.node.fun=FALSE,
    border.col=col,
    round=NULL, leaf.round=NULL,
    shadow.col=0, prefix="", suffix="", xsep=NULL,
```

```
under.percent=2, under.font=font, under.col=1, under.cex=.8,

split.cex=1, split.font=2, split.family=family, split.col=1,
split.box.col=0, split.border.col=0,
split.lty=1, split.lwd=NULL, split.round=0,
split.shadow.col=0,
split.prefix="", right.split.prefix=NULL,
split.suffix="", right.split.suffix=NULL,
facsep=",", eq=" = ", lt=" < ", ge=" >= ",

branch.col=if(is.zero(branch.type)) 1 else "gray",
branch.lty=1, branch.lwd=NULL,
branch.type=0, branch.tweak=1,
min.branch.width=.002, branch.fill=branch.col,

nn.cex=NULL, nn.font=3, nn.family="", nn.col=1,
nn.box.col=0, nn.border.col=nn.col,
nn.lty=1, nn.lwd=NULL, nn.round=.3,
yes.text="yes", no.text="no",

node.fun=NULL,
split.fun=NULL,
FUN="text",

nspace=branch, minbranch=.3, do.par=TRUE,
add.labs=TRUE, clip.left.labs=(type == 5), fam.main="",
yshift=0, yspace=space, shadow.offset=.4,
split.adj=NULL, split.yshift=0, split.space=space,
split.yspace=yspace, split.shadow.offset=shadow.offset,
nn.adj=.5, nn.yshift=0, nn.space=.8, nn.yspace=.5,
ygap=gap/2, under.ygap=.5, yesno.yshift=0,
xcompact=TRUE, ycompact=uniform, xcompact.ratio=.8, min.inter.height=4,
max.auto.cex=1, min.auto.cex=.15, ycompress.cex=.7, accept.cex=1.1,
shift.amounts=c(1.5, 2),
Fallen.yspace=.1, boxes.include.gap=FALSE,
legend.x=NULL, legend.y=NULL, legend.cex=1,
...)
```

Arguments

x	An rpart object. The only required argument.
type	Type of plot. Possible values: **0** Default. Draw a split label at each split and a node label at each leaf. **1** Label all nodes, not just leaves. Similar to text.rpart's all=TRUE. **2** Like 1 but draw the split labels below the node labels. Similar to the plots in the CART book. **3** Draw separate split labels for the left and right directions.

4 Like 3 but label all nodes, not just leaves. Similar to text.rpart's fancy=TRUE. See also clip.right.labs.

5 Show the split variable name in the interior nodes.

extra Display extra information at the nodes. Possible values:

"auto" (case insensitive)
Automatically select a value based on the model type, as follows:
extra=106 class model with a binary response
extra=104 class model with a response having more than two levels
extra=100 other models

0 Default. No extra information.

1 Display the number of observations that fall in the node (per class for class objects; prefixed by the number of events for poisson and exp models). Similar to text.rpart's use.n=TRUE.

2 Class models: display the classification rate at the node, expressed as the number of correct classifications and the number of observations in the node. Poisson and exp models: display the number of events.

3 Class models: misclassification rate at the node, expressed as the number of incorrect classifications and the number of observations in the node.

4 Class models: probability per class of observations in the node (conditioned on the node, sum across a node is 1).

5 Class models: like 4 but don't display the fitted class.

6 Class models: the probability of the second class only. Useful for binary responses.

7 Class models: like 6 but don't display the fitted class.

8 Class models: the probability of the fitted class.

9 Class models: The probability relative to *all* observations – the sum of these probabilities across all leaves is 1. This is in contrast to the options above, which give the probability relative to observations falling *in the node* – the sum of the probabilities across the node is 1.

10 Class models: Like 9 but display the probability of the second class only. Useful for binary responses.

11 Class models: Like 10 but don't display the fitted class.

+100 Add 100 to any of the above to also display the percentage of observations in the node. For example extra=101 displays the number and percentage of observations in the node. Actually, it's a weighted percentage using the weights passed to rpart.

Note: Unlike text.rpart, by default prp uses its own routine for generating node labels (not the function attached to the object). See the node.fun argument.

under — Applies only if extra > 0. Default FALSE, meaning put the extra text *in* the box. Use TRUE to put the text *under* the box. See also under.cex.

fallen.leaves — Default FALSE. If TRUE, position the leaf nodes at the bottom of the graph.

nn — Display the node numbers. Default FALSE. (In the current implementation some overplotting may occur with nn=TRUE.)

ni — Display the node indices, i.e. the row numbers of the nodes in the object's rpart.object frame. Default FALSE.

yesno — One of
0 don't write yes and no on the tree
1 (default) write yes and no at the top split
2 write yes and no at all splits. Seems to work best with fallen.leaves=TRUE, split.border.col=1.
(The yesno argument is ignored if type=3 or 4. Use nn.col and the other nn parameters to change the color etc. of the yes and no text. Use yes.text and no.text to change the actual text displayed.)

branch — Controls the shape of the branch lines. Specify a value between 0 (V shaped branches) and 1 (square shouldered branches).
Default is if(fallen.leaves) 1 else .2.

uniform — If TRUE (the default), the vertical spacing of the nodes is uniform. If FALSE, the nodes are spaced proportionally to the fit (more precisely, to the difference between a node's deviance and the sum of its two children's deviances). Very small vertical spaces are automatically artificially expanded to make room for the labels, see minbranch. Note: uniform=FALSE with cex=NULL (the default) can sometimes cause very small text.

left — Default TRUE, meaning the left side of a split is the path taken if the split condition is true. With left=FALSE the split labels are changed so the right side is true.

xflip — Default FALSE. If TRUE, flip the tree horizontally.

yflip — Default FALSE. If TRUE, flip the tree vertically, so the root is at the bottom.

digits — The number of significant digits in displayed numbers. Default 2.
If 0, use getOption("digits").
If negative, use the standard format function (with the absolute value of digits).

When digits is positive, the following details apply:
Numbers from 0.001 to 9999 are printed without an exponent (and the number of digits is actually only a suggestion, see format for details). Numbers out that range are printed with an "engineering" exponent (a multiple of 3).

varlen — Length of variable names in text at the splits (and, for class responses, the class in the node label). Default -8, meaning truncate to eight characters. Possible values:

0 use full names.

greater than 0 call abbreviate with the given varlen.

less than 0 truncate variable names to the shortest length where they are still unique, but never truncate to shorter than abs(varlen).

faclen — Length of factor level names in splits. Default 3, meaning abbreviate to three characters. Possible values are as varlen above, except that for back-compatibility with text.rpart the special value 1 means represent the factor levels with alphabetic characters (a for the first level, b for the second, etc.).

roundint — If roundint=TRUE (default) and all values of a predictor in the training data are integers, then splits for that predictor are rounded to integer. For example, display nsiblings < 3 instead of nsiblings < 2.5.
If roundint=TRUE and the data used to build the model is no longer available, a warning will be issued.
Using roundint=FALSE is advised if non-integer values are in fact possible for a predictor, even though all values in the training data for that predictor are integral.

cex — Default NULL, meaning calculate the text size automatically.
Since font sizes are discrete, the cex you ask for may not be exactly the cex you get.

tweak — Adjust the (possibly automatically calculated) cex. Using tweak is often easier than specifying cex.
The default tweak is 1, meaning no adjustment.
Use say tweak=1.2 to make the text 20% larger.
Since font sizes are discrete, a small change to tweak may not actually change the type size, or change it more than you want.

clip.facs — Default FALSE. If TRUE, print splits on factors as female instead of sex = female; the variable name and equals is dropped.
Another example: print survived or died rather than survived = survived or survived = died.

clip.right.labs — Applies only if type=3 or 4.
Default is TRUE meaning "clip" the right-hand split labels, i.e., don't print variable=.
See also clip.left.labs.

compress — If TRUE (the default), make more space by shifting nodes horizontally where space is available. This often allows larger text. (This is the same as plot.rpart's argument of the same name, except that here the default is TRUE.)

ycompress — If TRUE (the default unless uniform=FALSE), make more space by shifting labels vertically where space is available. Actually, this only kicks in if the initial automatically calculated cex is less than 0.7. Use ycompress=FALSE if you feel the resulting display is too messy. In the current implementation, the shifting algorithm works a little better (allowing larger text) with type=1, 2, or 3.

Margin — Extra white space around the tree, as a fraction of the graph width. Default 0, meaning no extra space. To add say 10% space around the tree use Margin=0.1. (This is the margin argument of plot.rpart. The name was changed to prevent partial matching with mar, which can be passed in as a ... argument.)

space — Horizontal space to the box border on each side of the node label text, in character widths. Default 1. Use this (and yspace) for bigger boxes. Since this affects the size of the (possibly invisible) boxes, it also affects the graph layout and hence also the automatic calculation of cex.

gap — Minimum horizontal gap between the (possibly invisible) boxes, in character widths. Default NULL, meaning automatically choose a suitable value (normally 1, but if the graph is very crowded will be set to 0, permitting boxes to touch to allow a bigger cex). See also space.

snip — Default FALSE. Set TRUE to interactively trim the tree with the mouse. See the package vignette (or just try it).

snip.fun — Function invoked after each mouse click when snip=TRUE. Default NULL, meaning no function. Otherwise set snip.fun to your own function with the prototype function(tree), where tree is the snipped tree. See the package vignette for an example.

The following control the node labels.

trace — Default FALSE. Use TRUE to print the automatically calculated cex, xlim, and ylim. Use integer values greater than 1 for more detailed tracing.

box.col — Color of the boxes around the text. Default 0, meaning use the background color. If this argument is used, the box.palette argument is ignored.

box.palette — Palette for coloring the node boxes based on the fitted value. This is a vector of colors, for example box.palette=c("green", "green2", "green4"). Small fitted values are displayed with colors at the start of the vector; large values with colors at the end. Quantiles are used to partition the fitted values.

The special value box.palette=0 (default for prp) uses the background color (typically white).

The special value box.palette="auto" (default for rpart.plot, case insensitive) automatically selects a predefined palette based on the type of model.

Otherwise specify a **predefined palette** e.g. box.palette="Grays" for the predefined gray palette (a range of grays). The predefined palettes are (see the show.prp.palettes function):
Grays Greys Greens Blues Browns Oranges Reds Purples
Gy Gn Bu Bn Or Rd Pu (alternative names for the above palettes)
BuGn GnRd BuOr etc. (two-color diverging palettes: any combination of two of the above palettes)
RdYlGn GnYlRd BlGnYl YlGnBl (three color palettes)

Prefix the palette name with "-" to reverse the order of the colors
e.g. box.palette="-auto" or box.palette="-Grays".

The box.palette argument is ignored if the box.col palette argument is specified.

pal.thresh — Applies when box.palette is a two-color diverging palette (such as BuGn). Specifies the response threshold to split the two sub-palettes (such as Bu and Gn).
For example, to display fitted values less than 90 in shades of blue and values greater than 90 in shades of green, use pal.thresh=90 with box.palette="BuGn". By default pal.thresh is calculated automatically. (For a two-class response

the default threshold is 0.5; for a continuous response the default is the median fitted value.)
Node boxes for fitted values less than the threshold are displayed using colors from the first sub-palette; boxes for fitted values greater than the threshold are displayed using colors from the second sub-palette.
This argument is ignored if box.palette isn't a two-color diverging palette, and is ignored for models with multiple-class responses (more than two classes).

pal.node.fun — Specifies how the box.palette argument is handled when the node.fun argument is specified.
Default is FALSE, meaning ignore node.fun and use the fitted value to select the node color from box.palette as usual.
If TRUE, use the label returned by the node function (instead of the fitted value) to select the color. The first number in each label returned by the node function is used, skipping over any non-numeric initial text in the label. An error message will be issued if a label doesn't include a number.
This argument is ignored if node.fun isn't specified, and is ignored for models with multiple-class responses (more than two classes).

border.col — Color of the box border around the text. Default col, the color of the text in the box. Use 0 for no border. (Note: par settings like col can be passed in as ... arguments. If not passed in, par("col") is used.)

round — Controls the rounding of the corners of the node boxes. Default NULL, meaning calculate automatically. Else specify 0 for sharp edges, and values greater than 0 for rounded edges. Bigger is more round. Values too big for the size of the box get silently reduced.

leaf.round — Controls the rounding of the corners of the leaf node boxes. Default NULL, meaning use round. Else specify a value greater than or equal to 0.

shadow.col — Color of the shadow under the boxes. Default 0, no shadow. Try "gray" or "darkgray". (Note: overlapping shadows look better on devices that support alpha channels. If you get the message "Warning: semi-transparency is not supported" please let me know – it means that a fix is needed to the code that determines if the device supports alpha channels.)

prefix — Default "". Prepend this string to the node labels. So could be the name of the fitted response, for instance.

suffix — Default "". Append this string to the node labels. Text after a double newline "\n\n" (if any) will be plotted *under* the box. (Actually, double newlines can be used in any of the prefix or suffix arguments for this purpose.)

xsep — String which separates the individual counts and probabilities in node labels when extra>0. Default NULL meaning automatically select: usually
" " (two spaces), but " / " for rates. Use xsep="/" for compatibility with text.rpart. See also facsep, which separates the factor levels in split labels.

under.percent — Control whitespace before the percentage (when 100 is used with the extra argument). One of
0 put a space before the percentage
1 put a newline before the percentage

2 (default) automatically choose a space or newline before the percentage.

The following control the text under the boxes (apply only if `under=TRUE` or there is a double newline `\n\n` in `prefix` or `suffix`).

`under.font` Font of the text under the box. Default `font` (which can be passed in as a ... argument).

`under.col` Color of the text under the box. Default `1`.

`under.cex` Size of the text under the box relative to the text in the box. Default `.8`, smaller than the text in the box.

The following control the split labels.

`split.cex` Size of the split text relative to `cex` (which by default is calculated automatically). Default `1`.

`split.font` Font for the split labels. Default `2`, bold. (Note: use `font` to change the *node* label text.)

`split.family` Font family for the split labels.
Default `""`, or use something like `split.family="serif"`. (Note: use `family` to change the *node* label text.)

`split.col` Color of the split label text. Default `1`. (Note: use `col` to change the *node* label text.)

`split.box.col` Color of the split boxes. Default `0`, meaning use the background color.

`split.border.col`
Color of the split box borders. Default `0`, invisible.

`split.lty` Line type for the split box borders. The default is `1`, but the border will be invisible unless you change the default `split.border.col`. (Note: use `lty` to change the *node* box borders.)

`split.lwd` Line width of the split box border relative to `cex` (which by default is calculated automatically). The border is by default invisible, see `split.border.col`.

`split.round` Controls the rounding of the corners of the split boxes. Default `0`, meaning sharp corners. Else specify a value greater than or equal to `0`. The split boxes are by default invisible, see `split.box.col` and `split.border.col`).

`split.shadow.col`
Color of the shadow under the split boxes. Default `0`, no shadow.

`split.prefix` Default `""`. Prepend this string to the split labels.

`right.split.prefix`
Default `split.prefix`. Prepend this string to the right split labels. Applies only when `type=3` or `4`.

`split.suffix` Default `""`. Append this string to the split labels.

`right.split.suffix`
Default `split.suffix`. Append this string to the right split labels. Applies only when `type=3` or `4`.

`facsep` Default `","`. String which separates the factor levels in split labels. See also `xsep`, which separates the individual counts when `extra` is used.

`eq` Default `" = "`. String which separates the factor name from the levels in split labels. The idea is that you can add or remove spaces around the `=`, or use words if that suits you.

`lt` Default `" < "`. String which represents "less than" in split labels.

`ge` Default `" >= "`. String which represents "greater than or equal" in split labels.

The following control the branches.

`branch.col` Color of the branch lines. Default `1`, but set to `"gray"` if `branch.type` is nonzero.

`branch.lty` Branch line type. Default `1`.

`branch.lwd` Line width of the branch lines relative to `cex` (which by default is calculated automatically). (Note: `branch.lwd` does not control the width of the "wide branches" drawn when `branch.type` is nonzero.)

`branch.type` Default `0`. If nonzero draw "wide branches", with branch widths proportional to the parameter selected by `branch.type` as follows:

0 The default. The branch lines are drawn conventionally.
1 deviance
2 sqrt(deviance)
3 deviance / nobs
4 sqrt(deviance / nobs) (the standard deviation when `method="anova"`)
5 weight (`frame$wt`). This is the number of observations at the node, unless `rpart`'s `weight` argument was used.
6 complexity
7 abs(predicted value)
8 predicted value - min(predicted value)
9 constant (for checking visual perception of the relative width of branches).

Otherwise set `branch.type` to your own function. The function should take a single argument `x` (the `rpart` object) and return a numeric vector of non-negative widths corresponding to rows in `frame`. See `get.branch.widths` in the source code.

Note: with a nonzero `branch.type`, in the current implementation the `branch` argument will be silently changed to `1` (if `branch > .5`) or `0` (if `branch < .5`)

`branch.tweak` Default `1`. Applies only if `branch.type` is nonzero. Use this argument to scale the widths of the branches, for example, `branch.tweak=.5` to halve the width of the branches. (By default, `prp` normalizes the widths so the widest branch is one-fifth the plot width.)

`min.branch.width`
Default `0.002`. Applies only if `branch.type` is nonzero. The minimum width of a branch, as a fraction of the page width. The width of branches that would be thinner than `min.branch.width` is clamped. Increase `min.branch.width` if the thinnest branches are too skinny on your display device.

`branch.fill` Color used to fill the wide branch lines. Applies only if `branch.type` is nonzero. Default `branch.col`.

The following control the node numbers (with nn=TRUE).

nn.cex — Default NULL, meaning calculate the cex of the node numbers automatically. This and the following arguments apply only when nn=TRUE.

nn.font — Font for the node numbers. Default 3, italic.

nn.family — Font family for the node numbers. Default "".

nn.col — Color of the node number text. Default 1.

nn.box.col — Color of the boxes around the node numbers. Default 0, meaning use the background color.

nn.border.col — Color of the box border around the node numbers. Default nn.col.

nn.lty — Line type of the node number box border. Default 1.

nn.lwd — Line width of the node box border relative to cex (which by default is calculated automatically). Default NULL, meaning use lwd (which can be passed in as a ... argument).

nn.round — Controls the rounding of the corners of the node number boxes. Default .3, meaning small corners. Else specify a value greater than or equal to 0.

yes.text, no.text

Text displayed when yesno=TRUE. Default yes.text="yes" and no.text="no".

The following are user definable functions.

node.fun — The function that generates the text at the node labels. The default is NULL, which means use a default function internal to prp. (This is necessary for full support of extra as described in the section on extra above.)
Otherwise set node.fun to your own function with the prototype
function(x, labs, digits, varlen)
See the package vignette for details. See also the pal.node.fun argument.

split.fun — The function that generates the text at the splits. The default is NULL, which means use a default function internal to prp. Otherwise set split.fun to your own function with the prototype function(x, labs, digits, varlen, faclen)

FUN — The function that displays the text on the screen. Default text.

The following are esoteric parameters, mostly for the graph layout engine.

nspace — Applies only when compress=TRUE. Default nspace=branch. The size of the space between a split and a leaf, relative to the space between leaves.

minbranch — Applies only when uniform=FALSE. Default .3. The minimum height between levels is clamped at minbranch times the mean interlevel distance. Needed because sometimes a split gives little or no improvement in deviance, and an interlevel distance strictly proportional to the improvement would leave no room for the label.

do.par — Default TRUE, meaning adjust the mar parameter so the tree fills the figure region. This also sets xpd=NA. These graphic parameters are restored to their original state before prp exits. If you explicitly set mar or xpd, prp will use your setting regardless of the setting of do.par.

add.labs — Default TRUE, meaning display the labels. If FALSE, gives a bare bones display similar to plot.rpart.

clip.left.labs Like clip.right.labs but for the left labels. Default is FALSE. Note that clip.left.labs and clip.right.labs can be vectors, indexed on the split number.

fam.main Font family for the main text. Default "". The (inconsistent) name was chosen to minimize partial matching with main and family which can be passed in as in as ... arguments.

yshift Vertical position of the labels, in character heights relative to their default position. Default 0. Negative values move the text down; positive up (the box around the text will follow along).

yspace Vertical space to the box border above and below the node label text, in character heights. Default space. See the comments for space.

shadow.offset Offset of the shadow from the boxes, in character widths. Default .4 (but the shadow will be invisible unless the default shadow.col is changed).

split.adj Horizontal position of the split text. In string width units, as is the convention for adj arguments. Default NULL, meaning use adj (which defaults to 0.5 but can be passed in as a ... argument). Use values less/more than .5 to shift the text left/right (the box around the text will follow along).

split.yshift Vertical position of the split labels, in character heights relative to their default positions. Default 0. Negative values move the text down; positive up (the box around the text will follow along). This adjusts the positions of the split labels relative to the node labels. (Use yshift if you want to shift *both* the split and node labels.)

split.space Horizontal space between the split label text and the box, in character widths. Default space. Affects the size of the box drawn around the text. The split boxes are by default invisible (see split.box.col and split.border.col), but nevertheless affect the graph layout used in the automatic calculation of cex.

split.yspace Vertical space between the split label text and the box, in character heights. Default yspace.

split.shadow.offset
Offset of the shadow from the split boxes, in character widths. Default shadow.offset. (but the shadow will be invisible unless the default shadow.col is changed).

nn.adj Horizontal position of the node label text. Default .5.

nn.yshift Vertical position of the node numbers, in character heights relative to their default positions. Default 0.

nn.space Horizontal space to the box border on each side of the node number text, in character widths. Default .8.

nn.yspace Vertical space to the box border above and below the node number text, in character heights. Default .5.

under.ygap Applies if text is plotted under the box (i.e. if under=TRUE or there is a double newline in prefix or suffix). Vertical gap (in char heights) between the lower edge of the box and the top of the text under the box.

yesno.yshift Vertical position of "yes" and "no" in character heights relative to their default position. Default 0. Applies only when yesno=TRUE.

ygap Minimum vertical gap between boxes, in character heights. Default gap/2.

xcompact — If TRUE (the default) and there is too much white space, automatically change xlim to compact the entire tree horizontally. This usually only activates for small trees. (The xcompact and ycompact arguments compact the tree as a whole, whereas the compress and ycompress arguments move parts of the tree into available space.)

ycompact — If TRUE (the default) and there is too much vertical space, automatically change ylim to compact the entire tree vertically.

xcompact.ratio — Default .8. Applies only when xcompact=TRUE. The maximum possible without overplotting is 1, but compacting by .8 usually gives more pleasing spacing (it gives more space).

min.inter.height — Default 4. Applies only when ycompact=TRUE. Minimum height (in units of character height) between the lowest label in a layer and the highest label in the layer below it.

max.auto.cex — Clamp the maximum automatically calculated cex at this value Default 1, meaning never expand cex, only contract when necessary.

min.auto.cex — Default .15. Never downscale to less than this when automatically calculating cex, even if overplotted labels result. (The graph layout algorithm is unstable with cex's below 0.15 – meaning that the automatic type size may be smaller than necessary.)

ycompress.cex — Default .7. Applies only when ycompress=TRUE. Apply the ycompress algorithm if the initial automatically calculated cex is less than this. The idea is that we don't want to shift if we get an acceptable cex without shifting. Make Inf to always attempt shifting.

accept.cex — Accept shifting only if it causes at least this much improvement in cex (because we don't want to shift if it gives only a small improvement in cex). Default 1.1 i.e. require at least a 10% improvement. Use 0 to always accept shifts and Inf to never accept (or use ycompress=FALSE).

shift.amounts — Default c(1.5, 2, 3). For ycompress, choose the best cex yielded by shifting nodes by these amounts, in multiples of the box heights (after initial scaling).

Fallen.yspace — Extra space for fallen leaves. Default .1, meaning allow 10% of the vertical space for the fallen leaves. (The name Fallen.yspace uses upper case to avoid partial matching with fallen.leaves.)

boxes.include.gap — Default FALSE. Include gap and ygap when drawing the boxes, for debugging purposes. (To draw the boxes, see box.col, border.col, split.box.col, and split.border.col.) This argument only affects the way the boxes are drawn, not the graph layout algorithm in any way. With the optimum cex at least one pair of boxes displayed in this manner will just touch (but none will overlap).

legend.x — Applies only to models with a multilevel class response (not binary or anova models, for which no legend is drawn).
Horizontal position of the legend. Typically a value between 0 and 1, although values beyond those limits are often useful.
Default is NULL meaning automatically position the legend (assuming there is enough space). Use NA for no legend.
Use trace=TRUE to see the automatically calculated legend position.

legend.y	Like legend.x but for the vertical position of the legend.
legend.cex	Like legend.x but for the relative size of the legend text. Default is 1.
...	Extra par arguments. Only the "important" par arguments are supported. Note that arguments like col apply only to the *node* labels. To affect the split labels or branch lines, use split.col and branch.col instead.

Value

A list with the following components. With the default args most of these are calculated automatically.

obj	The rpart object. Identical to the x argument passed in unless snip was used.
snipped.nodes	The snipped nodes, NULL unless snip was used.
xlim, ylim	The graph limits.
x, y	The node coords.
branch.x, branch.y	The branch line coords.
labs	The node labels.
cex	The node label cex.
boxes	The coords of the boxes around the nodes.
split.labs	The split labels.
split.cex	The split label cex.
split.boxes	The coords of the boxes around the splits.

See Also

The package vignette Plotting rpart trees with the rpart.plot package (also available here).

rpart.plot

Functions in the rpart package: plot.rpart text.rpart rpart

Examples

```
data(ptitanic)
tree <- rpart(survived ~ ., data = ptitanic, cp = .02)
                            # cp = .02 because want small tree for demo

old.par <- par(mfrow = c(2,2))
                            # put 4 figures on one page

prp(tree, main = "default prp\n(type =  0, extra = 0)")

prp(tree, main = "type = 4,  extra = 6\nbox.palette = \"auto\"",
    type = 4, extra = 6,    # label all nodes, show prob of second class
    box.palette = "auto",   # auto color the nodes based on the model type
    faclen = 0)             # faclen = 0 to print full factor names
```

```
cols <- ifelse(tree$frame$yval == 1, "darkred", "green4")
                          # green if survived

prp(tree, main = "assorted arguments",
    extra = 106,          # display prob of survival and percent of obs
    nn = TRUE,            # display the node numbers
    fallen.leaves = TRUE, # put the leaves on the bottom of the page
    shadow.col = "gray",  # shadows under the leaves
    branch.lty = 3,       # draw branches using dotted lines
    branch = .5,          # change angle of branch lines
    faclen = 0,           # faclen = 0 to print full factor names
    trace = 1,            # print the auto calculated cex, xlim, ylim
    split.cex = 1.2,      # make the split text larger than the node text
    split.prefix = "is ", # put "is " before split text
    split.suffix = "?",   # put "?" after split text
    col = cols, border.col = cols,   # green if survived
    split.box.col = "lightgray",   # lightgray split boxes (default is white)
    split.border.col = "darkgray", # darkgray border on split boxes
    split.round = .5)             # round the split box corners a tad

# compare to the plotting functions in the rpart package
plot(tree, uniform = TRUE, compress = TRUE, branch = .2)
text(tree, use.n = TRUE, cex = .8, xpd = NA) # cex is a guess, depends on your window size
title("compare to the plotting functions\nin the rpart package", cex.sub = .8)

par(old.par)
```

`ptitanic` *Titanic data with passenger names and other details removed.*

Description

Titanic data with passenger names and other details removed.

Format

A data frame with 1046 observations on 6 variables.

`pclass`	passenger class, unordered factor: 1st 2nd 3rd
`survived`	factor: died or survived
`sex`	unordered factor: male female
`age`	age in years, min 0.167 max 80.0
`sibsp`	number of siblings or spouses aboard, integer: 0...8
`parch`	number of parents or children aboard, integer: 0...6

Source

The dataset was compiled by Frank Harrell and Robert Dawson:
https://hbiostat.org/data/repo/titanic.html.

For this version of the Titanic data, passenger details were deleted, survived was cast as a factor, and the name changed to ptitanic to minimize confusion with other versions.

In this data the crew are conspicuous by their absence.

Contents of ptitanic:

```
      pclass survived    sex    age sibsp parch
1        1st survived female 29.000     0     0
2        1st survived   male  0.917     1     2
3        1st     died female  2.000     1     2
4        1st     died   male 30.000     1     2
5        1st     died female 25.000     1     2
...
1309     3rd     died   male 29.000     0     0
```

How ptitanic was built:

```
load("titanic3.sav") # from Dr. Harrell's web site
# discard name, ticket, fare, cabin, embarked, body, home.dest
ptitanic <- titanic3[,c(1,2,4,5,6,7)]
# change survived from integer to factor
ptitanic$survived <- factor(ptitanic$survived, labels = c("died", "survived"))
save(ptitanic, file = "ptitanic.rda")
```

This version of the data differs from etitanic in the earth package in that here survived is a factor (not an integer) and age has some NAs.

Examples

```
data(ptitanic)
summary(ptitanic)

# survival rate was greater for females
rpart.rules(rpart(survived ~ sex, data = ptitanic))

# survival rate was greater for higher classes
rpart.rules(rpart(survived ~ pclass, data = ptitanic))

# survival rate was greater for children
rpart.rules(rpart(survived ~ age, data = ptitanic))

# main indicator of missing data is 3rd class esp. with many children
obs.with.nas <- rowSums(is.na(ptitanic)) > 0
rpart.rules(rpart(obs.with.nas ~ ., data = ptitanic, method = "class"))
```

`rpart.plot` *Plot an rpart model. A simplified interface to the prp function.*

Description

Plot an `rpart` model, automatically tailoring the plot for the model's response type.

For an overview, please see the package vignette Plotting rpart trees with the rpart.plot package. (also available here).

This function is a simplified front-end to `prp`, with only the most useful arguments of that function, and with different defaults for some of the arguments. The different defaults mean that this function automatically creates a colored plot suitable for the type of model (whereas `prp` by default creates a minimal plot). See the `prp` help page for a table showing the different defaults.

Usage

```
rpart.plot(x = stop("no 'x' arg"),
    type = 2, extra = "auto",
    under = FALSE, fallen.leaves = TRUE,
    digits = 2, varlen = 0, faclen = 0, roundint = TRUE,
    cex = NULL, tweak = 1,
    clip.facs = FALSE, clip.right.labs = TRUE,
    snip = FALSE,
    box.palette = "auto", shadow.col = 0,
    ...)
```

Arguments

To start off, look at the arguments `x`, `type` and `extra`. Just those arguments will suffice for many users. If you don't want a colored plot, use `box.palette=0`.

`x` An `rpart` object. The only required argument.

`type` Type of plot. Possible values:

0 Draw a split label at each split and a node label at each leaf.

1 Label all nodes, not just leaves. Similar to `text.rpart`'s `all=TRUE`.

2 Default. Like `1` but draw the split labels below the node labels. Similar to the plots in the CART book.

3 Draw separate split labels for the left and right directions.

4 Like `3` but label all nodes, not just leaves. Similar to `text.rpart`'s `fancy=TRUE`. See also `clip.right.labs`.

5 Show the split variable name in the interior nodes.

`extra` Display extra information at the nodes. Possible values:

"auto" (case insensitive) Default.
Automatically select a value based on the model type, as follows:

`extra=106` class model with a binary response
`extra=104` class model with a response having more than two levels
`extra=100` other models

0 No extra information.

1 Display the number of observations that fall in the node (per class for `class` objects; prefixed by the number of events for `poisson` and `exp` models). Similar to `text.rpart`'s `use.n=TRUE`.

2 Class models: display the classification rate at the node, expressed as the number of correct classifications and the number of observations in the node. Poisson and exp models: display the number of events.

3 Class models: misclassification rate at the node, expressed as the number of incorrect classifications and the number of observations in the node.

4 Class models: probability per class of observations in the node (conditioned on the node, sum across a node is 1).

5 Class models: like `4` but don't display the fitted class.

6 Class models: the probability of the second class only. Useful for binary responses.

7 Class models: like `6` but don't display the fitted class.

8 Class models: the probability of the fitted class.

9 Class models: The probability relative to *all* observations – the sum of these probabilities across all leaves is 1. This is in contrast to the options above, which give the probability relative to observations falling *in the node* – the sum of the probabilities across the node is 1.

10 Class models: Like `9` but display the probability of the second class only. Useful for binary responses.

11 Class models: Like `10` but don't display the fitted class.

+100 Add `100` to any of the above to also display the percentage of observations in the node. For example `extra=101` displays the number and percentage of observations in the node. Actually, it's a weighted percentage using the `weights` passed to `rpart`.

Note: Unlike `text.rpart`, by default `prp` uses its own routine for generating node labels (not the function attached to the object). See the `node.fun` argument of `prp`.

`under` Applies only if `extra > 0`. Default `FALSE`, meaning put the extra text *in* the box. Use `TRUE` to put the text *under* the box.

`fallen.leaves` Default `TRUE` to position the leaf nodes at the bottom of the graph. It can be helpful to use `FALSE` if the graph is too crowded and the text size is too small.

`digits` The number of significant digits in displayed numbers. Default `2`.
If `0`, use `getOption("digits")`.
If negative, use the standard `format` function (with the absolute value of `digits`).

When digits is positive, the following details apply:
Numbers from 0.001 to 9999 are printed without an exponent (and the number of digits is actually only a suggestion, see format for details). Numbers out that range are printed with an "engineering" exponent (a multiple of 3).

varlen — Length of variable names in text at the splits (and, for class responses, the class in the node label). Default 0, meaning display the full variable names. Possible values:

0 use full names (default).

greater than 0 call abbreviate with the given varlen.

less than 0 truncate variable names to the shortest length where they are still unique, but never truncate to shorter than abs(varlen).

faclen — Length of factor level names in splits. Default 0, meaning display the full factor names. Possible values are as varlen above, except that for back-compatibility with text.rpart the special value 1 means represent the factor levels with alphabetic characters (a for the first level, b for the second, etc.).

roundint — If roundint=TRUE (default) and all values of a predictor in the training data are integers, then splits for that predictor are rounded to integer. For example, display nsiblings < 3 instead of nsiblings < 2.5.
If roundint=TRUE and the data used to build the model is no longer available, a warning will be issued.
Using roundint=FALSE is advised if non-integer values are in fact possible for a predictor, even though all values in the training data for that predictor are integral.

cex — Default NULL, meaning calculate the text size automatically.
Since font sizes are discrete, the cex you ask for may not be exactly the cex you get.

tweak — Adjust the (possibly automatically calculated) cex. Using tweak is often easier than specifying cex.
The default tweak is 1, meaning no adjustment.
Use say tweak=1.2 to make the text 20% larger.
Since font sizes are discrete, a small change to tweak may not actually change the type size, or change it more than you want.

clip.facs — Default FALSE. If TRUE, print splits on factors as female instead of sex = female; the variable name and equals is dropped.
Another example: print survived or died rather than survived = survived or survived = died.

clip.right.labs
Applies only if type=3 or 4.
Default is TRUE meaning "clip" the right-hand split labels, i.e., don't print variable=.

snip — Default FALSE. Set TRUE to interactively trim the tree with the mouse. See the package vignette (or just try it).

`box.palette` Palette for coloring the node boxes based on the fitted value. This is a vector of `colors`, for example `box.palette=c("green", "green2", "green4")`. Small fitted values are displayed with colors at the start of the vector; large values with colors at the end. Quantiles are used to partition the fitted values.

The special value `box.palette=0` (default for `prp`) uses the background color (typically white).

The special value `box.palette="auto"` (default for `rpart.plot`, case insensitive) automatically selects a predefined palette based on the type of model.

Otherwise specify a **predefined palette** e.g. `box.palette="Grays"` for the predefined gray palette (a range of grays). The predefined palettes are (see the `show.prp.palettes` function):
`Grays Greys Greens Blues Browns Oranges Reds Purples`
`Gy Gn Bu Bn Or Rd Pu` (alternative names for the above palettes)
`BuGn GnRd BuOr` etc. (two-color diverging palettes: any combination of two of the above palettes)
`RdYlGn GnYlRd BlGnYl YlGnBl` (three color palettes)

Prefix the palette name with `"-"` to reverse the order of the colors
e.g. `box.palette="-auto"` or `box.palette="-Grays"`.

`shadow.col` Color of the shadow under the boxes. Default `0`, no shadow. Try `"gray"` or `"darkgray"`.

`...` Extra arguments passed to `prp` and the plotting routines. Any of `prp`'s arguments can be used.

Value

The returned value is identical to that of `prp`.

Author(s)

Stephen Milborrow, borrowing heavily from the `rpart` package by Terry M. Therneau and Beth Atkinson, and the R port of that package by Brian Ripley.

See Also

The package vignette Plotting rpart trees with the rpart.plot package (also available here).
`prp`
`rpart.rules`
Functions in the `rpart` package: `plot.rpart` `text.rpart` `rpart`

Examples

```
old.par <- par(mfrow=c(2,2))            # put 4 figures on one page

data(ptitanic)

#--------------------------------------------------------------------------

binary.model <- rpart(survived ~ ., data = ptitanic, cp = .02)
                                        # cp = .02 for small demo tree
```

```
rpart.plot(binary.model,
           main = "titanic survived\n(binary response)")

rpart.plot(binary.model, type = 3, clip.right.labs = FALSE,
           branch = .4,
           box.palette = "Grays",       # override default GnBu palette
           main = "type = 3, clip.right.labs = FALSE, ...\n")

#-------------------------------------------------------------------------

anova.model <- rpart(Mileage ~ ., data = cu.summary)

rpart.plot(anova.model,
           shadow.col = "gray",         # add shadows just for kicks
           main = "miles per gallon\n(continuous response)\n")

#-------------------------------------------------------------------------

multi.class.model <- rpart(Reliability ~ ., data = cu.summary)

rpart.plot(multi.class.model,
           main = "vehicle reliability\n(multi class response)")

par(old.par)
```

`rpart.predict` *Extended version of predict.rpart*

Description

Identical to `predict.rpart` but optionally show the node numbers and rules for the predicted values.

Usage

```
rpart.predict(object, newdata,
              type = c("vector", "prob", "class", "matrix"),
              na.action = na.pass,
              nn=FALSE, rules=FALSE, ...)
```

Arguments

`object`, `newdata`, `type`, `na.action`	Identical to the same arguments for `predict.rpart`. If both `nn` and `rules` are `FALSE`, the returned value is identical to `predict.rpart`.
`nn`	If `TRUE`, return a `data.frame` with the predictions as usual but with an extra column showing the leaf node number for each prediction.

rules	If TRUE, return a data.frame with the predictions as usual but with an extra column showing the rpart rule (as a string) for each prediction. It may be helpful to use options(width=1000) before printing this data.frame.
...	Passed on to rpart.rules, for example clip.facs=TRUE.

Value

Same as predict.rpart, but with additional information if nn=TRUE and/or rules=TRUE.

See Also

predict.rpart
rpart.rules

Examples

```
data(ptitanic)
model <- rpart(survived ~ ., data = ptitanic, cp = .02)
head(rpart.predict(model, rules=TRUE))
```

---

rpart.rules *Print an rpart model as a set of rules.*

Description

Print an rpart model as a set of rules.

Usage

```
rpart.rules(x = stop("no 'x' argument"),
            style = "wide", cover = FALSE, nn = FALSE,
            roundint = TRUE, clip.facs = FALSE,
            varorder = NULL, ...)

## S3 method for class 'rpart.rules'
print(x = stop("no 'x' argument"), style = attr(x, "style"), ...)
```

Arguments

x	An rpart object. The only required argument.
style	One of: **"wide"** (default) One rule per line. May require a lot of horizontal space. **"tall"** One split per line. **"tallw"** Like "tall" but with more horizontal white space for readability.
cover	Default FALSE. If TRUE, also print the percentage of cases covered by each rule.
nn	Default FALSE. If TRUE, also print the leaf node number for each rule.

`roundint` If `roundint=TRUE` (default) and all values of a predictor in the training data are integers, then splits for that predictor are rounded to integer. For example, display `nsiblings < 3` instead of `nsiblings < 2.5`.
Identical to the argument of the same name in `rpart.plot`, see there for details.

`clip.facs` Default `FALSE`. If `TRUE`, print splits on factors as `female` instead of `sex = female`; the variable name and equals is dropped.
Identical to the argument of the same name in `rpart.plot`.

`varorder` By default, the variables in the rules are ordered left to right on importance, where the "importance" of a variable here is the number of rules it appears in.
Use `varorder` to force variables to appear first in the rules. For example `varorder="sex"` or `varorder=c("sex", "pclass")` will put the specified variables first.
Partial matching of variable names is supported.

`...` The following can be passed as dot arguments. See `prp` for details on these arguments.

argument	**default**	
`extra`	`= "auto"`	a subset of the legal values for `prp` are supported
`digits`	`= 2`	default is two digits of accuracy, increase if necessary
`varlen`	`= 0`	default displays full variable names
`faclen`	`= 0`	default displays full factor names
`trace`	`= 0`	
`facsep`	`= " or "`	
`eq`	`= " is "`	
`lt`	`= " < "`	
`ge`	`= " >= "`	
`and`	`= " & "`	
`when`	`= " when "`	
`because`	`= " because "`	used only by `rpart.predict`
`null.model`	`= "null model"`	for root-only models (no splits)
`response.name`	`= NULL`	the response name printed before the rules (`NULL` means automatic)

Value

A `data.frame` of class `c("rpart.rules", "data.frame")` with some attached attributes which are passed on to `print.rpart.rules`. Note that `print.rpart.rules` temporarily increases `options(width)`.

See Also

`rpart.plot`

Examples

```
data(ptitanic)
model <- rpart(survived ~ ., data = ptitanic, cp = .02)
rpart.plot(model)
rpart.rules(model)
```

show.prp.palettes *Show the built-in prp palettes.*

Description

Display a diagram showing the built-in palettes accepted by prp's box.palette argument.

Usage

```
show.prp.palettes()
```

Examples

```
show.prp.palettes()
```

Index

∗ **CART**
  prp, 2
  rpart.plot, 17
∗ **datasets**
  ptitanic, 15
∗ **partitioning**
  prp, 2
  rpart.plot, 17
∗ **recursive**
  prp, 2
  rpart.plot, 17
∗ **rpart**
  prp, 2
  rpart.plot, 17
∗ **tree**
  prp, 2
  rpart.plot, 17

abbreviate, *6*, *19*
alpha, *8*

colors, *7*, *20*

earth, *16*
etitanic, *16*

format, *5*, *18*, *19*

getOption, *5*, *18*

par, *14*
plot.rpart, *14*, *20*
predict.rpart, *21*, *22*
print.rpart.rules *(*rpart.rules*)*, 22
prp, *2*, 2, *17*, *20*, *23*
ptitanic, 15

rpart, *2*, *3*, *14*, *17*, *20*, *22*
rpart.object, *5*
rpart.plot, *2*, *14*, 17, *23*
rpart.predict, 21, *23*
rpart.rules, *20*, *22*, 22

show.prp.palettes, *7*, *20*, 24

text, *11*
text.rpart, *6*, *14*, *19*, *20*