General Tree Structures

Description

Class "dendrogram" provides general functions for handling tree-like structures. It is intended as a replacement for similar functions in hierarchical clustering and classification/regression trees, such that all of these can use the same engine for plotting or cutting trees.

Usage

```
as.dendrogram(object, ...)
## S3 method for class 'hclust'
as.dendrogram(object, hang = -1, check = TRUE, ...)

## S3 method for class 'dendrogram'
as.hclust(x, ...)

## S3 method for class 'dendrogram'
plot(x, type = c("rectangle", "triangle"),
      center = FALSE,
      edge.root = is.leaf(x) || !is.null(attr(x,"edgetext")),
      nodePar = NULL, edgePar = list(),
      leaflab = c("perpendicular", "textlike", "none"),
      dLeaf = NULL, xlab = "", ylab = "", xaxt = "n", yaxt = "s",
      horiz = FALSE, frame.plot = FALSE, xlim, ylim, ...)

## S3 method for class 'dendrogram'
cut(x, h, ...)

## S3 method for class 'dendrogram'
merge(x, y, ..., height,
      adjust = c("auto", "add.max", "none"))

## S3 method for class 'dendrogram'
nobs(object, ...)

## S3 method for class 'dendrogram'
print(x, digits, ...)

## S3 method for class 'dendrogram'
rev(x)

## S3 method for class 'dendrogram'
str(object, max.level = NA, digits.d = 3,
    give.attr = FALSE, wid = getOption("width"),
    nest.lev = 0, indent.str = "",
    last.str = getOption("str.dendrogram.last"), stem = "--",
    ...)

is.leaf(object)
```

Arguments

object	any R object that can be made into one of class "dendrogram".
x, y	object(s) of class "dendrogram".
hang	numeric scalar indicating how the *height* of leaves should be computed from the heights of their parents; see plot.hclust.
check	logical indicating if object should be checked for validity. This check is not necessary when x is known to be valid such as when it is the direct result ofhclust(). The default is check=TRUE, e.g. for protecting against memory explosion with invalid inputs.
type	type of plot.
center	logical; if TRUE, nodes are plotted centered with respect to the leaves in the branch. Otherwise (default), plot them in the middle of all direct child nodes.
edge.root	logical; if true, draw an edge to the root node.
nodePar	a list of plotting parameters to use for the nodes (see points) or NULL by default which does not draw symbols at the nodes. The list may contain components namedpch, cex, col, xpd, and/or bg each of which can have length two for specifying separate attributes for *inner* nodes and *leaves*. Note that the default of pch is 1:2, so you may want to use pch = NA if you specify nodePar.
edgePar	a list of plotting parameters to use for the edge segments and labels (if there's anedgetext). The list may contain components named col, lty and lwd (for the segments), p.col, p.lwd, and p.lty (for the polygon around the text) andt.col for the text color. As with nodePar, each can have length two for differentiating leaves and inner nodes.
leaflab	a string specifying how leaves are labeled. The default "perpendicular" write text vertically (by default). "textlike" writes text horizontally (in a rectangle), and "none" suppresses leaf labels.
dLeaf	a number specifying the **d**istance in user coordinates between the tip of a leaf and its label. If NULL as per default, 3/4 of a letter width or height is used.
horiz	logical indicating if the dendrogram should be drawn *horizontally* or not.
frame.plot	logical indicating if a box around the plot should be drawn, see plot.default.
h	height at which the tree is cut.
height	height at which the two dendrograms should be merged. If not specified (or NULL), the default is ten percent larger than the (larger of the) two component heights.
adjust	a string determining if the leaf values should be adjusted. The default, "auto", checks if the (first) two dendrograms both start at 1; if they do, code"add.max" is

	chosen, which adds the maximum of the previous dendrogram leaf values to each leaf of the “next” dendrogram. Specifying `adjust` to another value skips the check and hence is a tad more efficient.
`xlim, ylim`	optional x- and y-limits of the plot, passed to `plot.default`. The defaults for these show the full dendrogram.
`..., xlab, ylab, xaxt, yaxt`	graphical parameters, or arguments for other methods.
`digits`	integer specifying the precision for printing, see `print.default`.
`max.level, digits.d, give.attr, wid, nest.lev, indent.str`	arguments to `str`, see `str.default()`. Note that `give.attr = FALSE` still shows `height` and `members` attributes for each node.
`last.str, stem`	strings used for `str()` specifying how the last branch (at each level) should start and the *stem* to use for each dendrogram branch. In some environments, using`last.str = "'"` will provide much nicer looking output, than the historical default`last.str = "`"`.

Details

The dendrogram is directly represented as a nested list where each component corresponds to a branch of the tree. Hence, the first branch of tree `z` is `z[[1]]`, the second branch of the corresponding subtree is `z[[1]][[2]]`, or shorter `z[[c(1,2)]]`, etc.. Each node of the tree carries some information needed for efficient plotting or cutting as attributes, of which only `members`, `height` and`leaf` for leaves are compulsory:

`members`

: total number of leaves in the branch

`height`

: numeric non-negative height at which the node is plotted.

`midpoint`

: numeric horizontal distance of the node from the left border (the leftmost leaf) of the branch (unit 1 between all leaves). This is used for `plot(*, center = FALSE)`.

`label`

: character; the label of the node

`x.member`

for `cut()$upper`, the number of *former* members; more generally a substitute for the`members` component used for 'horizontal' (when `horiz = FALSE`, else 'vertical') alignment.

`edgetext`

character; the label for the edge leading to the node

`nodePar`

a named list (of length-1 components) specifying node-specific attributes for `points` plotting, see the `nodePar` argument above.

`edgePar`

a named list (of length-1 components) specifying attributes for `segments` plotting of the edge leading to the node, and drawing of the `edgetext` if available, see the `edgePar` argument above.

`leaf`

logical, if `TRUE`, the node is a leaf of the tree.

`cut.dendrogram()` returns a list with components `$upper` and `$lower`, the first is a truncated version of the original tree, also of class `dendrogram`, the latter a list with the branches obtained from cutting the tree, each a `dendrogram`.

There are `[[`, `print`, and `str` methods for `"dendrogram"` objects where the first one (extraction) ensures that selecting sub-branches keeps the class, i.e., returns a dendrogram even if only a leaf. On the other hand, `[` (*single* bracket) extraction returns the underlying list structure.

Objects of class `"hclust"` can be converted to class `"dendrogram"` using method`as.dendrogram()`, and since R 2.13.0, there is also a `as.hclust()` method as an inverse.

`rev.dendrogram` simply returns the dendrogram `x` with reversed nodes, see also`reorder.dendrogram`.

The `merge(x, y, ...)` method merges two or more dendrograms into a new one which has `x` and`y` (and optional further arguments) as branches. Note that before **R** 3.1.2, `adjust = "none"` was used implicitly, which is invalid when, e.g., the dendrograms are from`as.dendrogram(hclust(..))`.

`nobs(object)` returns the total number of leaves (the `members` attribute, see above).

`is.leaf(object)` returns logical indicating if `object` is a leaf (the most simple dendrogram).

`plotNode()` and `plotNodeLimit()` are helper functions.

Warning

Some operations on dendrograms such as `merge()` make use of recursion. For deep trees it may be necessary to increase `options("expressions")`: if you do, you are likely to need to set the C stack size (`Cstack_info()[["size"]]`) larger than the default where possible.

Note

`plot()`:

> When using `type = "triangle"`, `center = TRUE` often looks better.

`str(d)`:

> If you really want to see the *internal* structure, use `str(unclass(d))` instead.

See Also

`dendrapply` for applying a function to *each* node. `order.dendrogram` and `reorder.dendrogram`; further, the `labels` method.

Examples

```
require(graphics); require(utils)

hc <- hclust(dist(USArrests), "ave")
(dend1 <- as.dendrogram(hc)) # "print()" method
str(dend1)          # "str()" method
str(dend1, max = 2, last.str =  "'") # only the first two sub-levels
oo <- options(str.dendrogram.last = "\\") # yet another possibility
str(dend1, max = 2) # only the first two sub-levels
options(oo)  # .. resetting them

op <- par(mfrow =  c(2,2), mar = c(5,2,1,4))
plot(dend1)
## "triangle" type and show inner nodes:
plot(dend1, nodePar = list(pch = c(1,NA), cex = 0.8, lab.cex = 0.8),
      type = "t", center = TRUE)
plot(dend1, edgePar = list(col = 1:2, lty = 2:3),
     dLeaf = 1, edge.root = TRUE)
plot(dend1, nodePar = list(pch = 2:1, cex = .4*2:1, col = 2:3),
     horiz = TRUE)

## simple test for as.hclust() as the inverse of as.dendrogram():
stopifnot(identical(as.hclust(dend1)[1:4], hc[1:4]))

dend2 <- cut(dend1, h = 70)
plot(dend2$upper)
## leaves are wrong horizontally:
plot(dend2$upper, nodePar = list(pch = c(1,7), col = 2:1))
##  dend2$lower is *NOT* a dendrogram, but a list of .. :
```

```
plot(dend2$lower[[3]], nodePar = list(col = 4), horiz = TRUE, type = "tr")
## "inner" and "leaf" edges in different type & color :
plot(dend2$lower[[2]], nodePar = list(col = 1),   # non empty list
     edgePar = list(lty = 1:2, col = 2:1), edge.root = TRUE)
par(op)
d3 <- dend2$lower[[2]][[2]][[1]]
stopifnot(identical(d3, dend2$lower[[2]][[c(2,1)]]))
str(d3, last.str = "'")

## to peek at the inner structure "if you must", use '[..]' indexing :
str(d3[2][[1]]) ## or the full
str(d3[])

## merge() to join dendrograms:
(d13 <- merge(dend2$lower[[1]], dend2$lower[[3]]))
## merge() all parts back (using default 'height' instead of original one):
den.1 <- Reduce(merge, dend2$lower)
## or merge() all four parts at same height --> 4 branches (!)
d. <- merge(dend2$lower[[1]], dend2$lower[[2]], dend2$lower[[3]],
            dend2$lower[[4]])
## (with a warning) or the same using  do.call :
stopifnot(identical(d., do.call(merge, dend2$lower)))
plot(d., main = "merge(d1, d2, d3, d4)  |->  dendrogram with a 4-split")

## "Zoom" in to the first dendrogram :
plot(dend1, xlim = c(1,20), ylim = c(1,50))

nP <- list(col = 3:2, cex = c(2.0, 0.75), pch =  21:22,
           bg =  c("light blue", "pink"),
           lab.cex = 0.75, lab.col = "tomato")
plot(d3, nodePar= nP, edgePar = list(col = "gray", lwd = 2), horiz = TRUE)

addE <- function(n) {
      if(!is.leaf(n)) {
        attr(n, "edgePar") <- list(p.col = "plum")
        attr(n, "edgetext") <- paste(attr(n,"members"),"members")
      }
      n
}
d3e <- dendrapply(d3, addE)
plot(d3e, nodePar =  nP)
plot(d3e, nodePar =  nP, leaflab = "textlike")
```