Package: Heatplus (via r-universe)

October 19, 2026

Version 3.20.0

Date 2020-03-01

Title Heatmaps with row and/or column covariates and colored clusters

Author Alexander Ploner <Alexander.Ploner@ki.se>

Maintainer Alexander Ploner <Alexander.Ploner@ki.se>

Description Display a rectangular heatmap (intensity plot) of a data matrix. By default, both samples (columns) and features (row) of the matrix are sorted according to a hierarchical clustering, and the corresponding dendrogram is plotted. Optionally, panels with additional information about samples and features can be added to the plot.

Imports graphics, grDevices, stats, RColorBrewer

Suggests Biobase, hgu95av2.db, limma

biocViews Microarray, Visualization

License GPL (>= 2)

URL https://github.com/alexploner/Heatplus

BugReports https://github.com/alexploner/Heatplus/issues

RoxygenNote 7.1.1

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:31:42 UTC

RemoteUrl https://github.com/bioc/Heatplus

RemoteRef RELEASE_3_23

RemoteSha 7492bc9d41824d77d4150f836a2de4f9ca4793bb

Contents

annHeatmap 2
annHeatmap2 4
breakColors 6

BrewerClusterCol 8
convAnnData 9
doLegend 10
g2r.colors 11
heatmap_2 12
heatmap_plus 14
heatmapLayout 17
modifyExistingList 18
niceBreaks 20
oldCutplot.dendrogram 21
oldPicketplot 22
picketPlot 24
picketPlotControl 25
plot.annHeatmap 27
print.annHeatmap 28
regHeatmap 29
RGBColVec 30

Index **32**

`annHeatmap` *Annotated heatmaps*

Description

Creating heatmaps with annotated columns

Usage

```
annHeatmap(x, ...)

## Default S3 method:
annHeatmap(
  x,
  annotation,
 dendrogram = list(clustfun = hclust, distfun = dist, Col = list(status = "yes"), Row
    = list(status = "hidden")),
  cluster = NULL,
  labels = NULL,
  legend = TRUE,
  ...
)

## S3 method for class 'ExpressionSet'
annHeatmap(x, ...)
```

Arguments

x	either a numerical matrix with the data for the central heatmap (for the default method) or an object of class ExpressionSet
...	extra options passed to annHeatmap2
annotation	a data frame containing the annotation for the columns of x
dendrogram	a list controlling the options for row- and column dendrogram, see annHeatmap2
cluster	a list controlling the options for clustering rows and columns of x, see annHeatmap2
labels	a list controlling the row- and column labels as well as their location and size, see annHeatmap2
legend	either a logical value, indicating whether to draw a legend at the default location determined by the function, or one of the sides of the plot (1-4), see annHeatmap2

Details

These functions generate an object representing the heatmap; in order to produce graphical output, you have to invoke the plot method, see Examples.

Value

An object of class annHeatmap

Warning

These are currently simple convenience functions that allow quick plotting, but little control over the finer details. This may change in the future, but for now, if you want to do anything fancy, you should invoke annHeatmap2 directly.

See Also

annHeatmap2, plot.annHeatmap

Examples

```
## Default method
set.seed(219)
mat = matrix(rnorm(100), ncol=5)
ann = data.frame(Class=c("A","A","B","A","B"))
map1 = annHeatmap(mat, ann)
plot(map1)

## Expression set
require(Biobase)
data(sample.ExpressionSet)
map2 = annHeatmap(sample.ExpressionSet)
plot(map2)
```

annHeatmap2 *Annotated heatmaps*

Description

This function plots a data matrix as intensity heatmap, with optional dendrograms, annotation panels and clustering for both rows and columns. This is the actual working function called by numerous wrappers.

Usage

```
annHeatmap2(
  x,
  dendrogram,
  annotation,
  cluster,
  labels,
  scale = c("row", "col", "none"),
  breaks = 256,
  col = g2r.colors,
  legend = FALSE
)
```

Arguments

x	the numerical matrix to be shown as heatmap
dendrogram	a list that controls how row- and column diagrams are determined and displayed
annotation	a list that controls the data and the way it is shown in row- and column annotation panels
cluster	a list that controls how many clusters are chosen, and how these clusters are labelled and colored
labels	a list that controls the row- and column labels, as well as their size and placement
scale	a character string indicating how the matrix x is standardized (by row, by column or not at all). This affects only display, not dendrograms or clustering
breaks	specifies the interval breaks for displaying the data in x; either a vector of explicit interval breaks, or just the desired number of intervals. See niceBreaks for details.
col	specifies a palette of colors for the heatmap intensities; either a vector of explicit color definitions (one less than breaks) or a palette function. See breakColors.
legend	whether and where to draw a legend for the colors/intervals in the heatmap. If TRUE, a legend is placed in a position determined by the function to be suitable; alternatively, integer values 1-4 indicate the side where the legend is to be drawn; and FALSE indicates that no legend should be drawn.

Details

Arguments scale, breaks, col and legend control different aspects of the whole plot directly as described. Arguments dendrogram, annotation, cluster and labels control aspects that may differ for the rows and columns of the central heatmap and have a special structure: each is a named list with different entries controling e.g. the look of a dendrogram, the data for annotation etc. Additionally, they can contain two extra entries called simply Row and Col; these are again named lists that can contain all the same entries as the parent list. Entries specified directly in the list apply to both rows and columns; entries specified as part of Row or Col override these defaults for the rows or columns only.

Recognized parameters for argument dendrogram:

clustfun the clustering function for generating the dendrogram; defaults to hclust for rows and columns

distfun a function that returns the pairwise distances between samples/features as an object of class dist; defaults to dist for rows and columns

status a string that controls the display of the dendrogram: yes means use the dendrogram to re-order the rows/columns and display the dendrogram; hidden means re-order, but do not display; no means do not use the dendrogram at all.

lwd the line width of the branches of the dendrogram; defaults to 3.

dendro an override argument that allows to pass in a dendrogram directly, bypassing the clustfun and distfun mechanism; defaults to NULL (i.e. is not used)

Recognized entries for argument annotation:

data a data frame containing the annotation data; defaults to NULL, i.e. no annotation is displayed

control a list of fine-tuning parameters that is passed directly to picketPlot; defaults to an empty list, i.e. the default settings in picketPlot

asIs logical value indicating whether the annotation data needs to be pre-processed via convAnnData or not; defaults to TRUE

inclRef logical value indicating whether to include all levels of factor variables in data, or whether to drop the reference level (i.e. the first level). Defaults to TRUE

Recognized entries for argument cluster:

cuth the height at which to cut through the dendrogram to define groups of similar features/samples; defaults to NULL, i.e. no cutting

label labels for the clusters; defaults to NULL, i.e. no labels

col colors for the different clusters; the colors are used for coloring both the sub-trees of the dendrogram and the corresponding area in the annotation plot (if there is one). This is either a vector of colors, or a palette function that takes a number and returns a vector of colors of the specified length; defaults to BrewerClusterCol

grp an override argument that directly specifies group memberships for the features/samples, completely bypassing the whole dendrogram and cuth mechanism. This probably only works for dendrogram$status="no".

Recognized entries for argument labels:

cex size of the text for the labels; defaults to NULL, i.e. use a hard-coded default guess

nrow amount of space available for the labels between the central heatmap and the dendrogram, expressed as lines of text; defaults to 3.

side side at which to draw the labels, coded as integer between 1 and 4 in the usual way (1 = below the plot, continuing clockwise). A common default for rows and columns does not make sense: rows only work with 2 and 4, columns only with 1 and 3. Defaults try to make use of empty space, depending on the presence of a dendrogram.

labels labels for rows and columns; defaults to NULL, i.e. using the row- and column names of x. Note that these labels are applied *after* re-sorting rows and columns as per dendrogram, so these have to be already sorted accordingly. If you want to change the labels *before* re-sorting, it is is easier to re-set the row- and/or column names of x.

Value

An object of class annHeatmap. Use plot to display it graphically.

See Also

heatmapLayout, niceBreaks, breakColors, g2r.colors, BrewerClusterCol

Examples

```
require(Biobase)
data(sample.ExpressionSet)
ex1 = sample.ExpressionSet[51:85,]
map1 = annHeatmap2(exprs(ex1), ann=list(Col=list(data=pData(ex1))),
                   cluster=list(Col=list(cuth=3000)))
plot(map1)
```

---

breakColors *Color palette for (symmetric) breaks*

Description

Given a vector of breaks specifying a set of intervals, this function provides a vector of colors for the indicating the intervals graphically. If the intervals are arranged symmetrically around a specified value, the colors try to reflect this.

Usage

```
breakColors(breaks, colors, center = 0, tol = 0.001)
```

Arguments

breaks	a vector of breaks
colors	either an explicit vector of colors, or a palette function that takes a number and returns a vector of colors
center	optional center around which to check for symmetry
tol	tolerance (as relative error) for deviation from mathematically exact symmetry

Details

The meaning of symmetrical is rather generous here: it is enough that the intervals specified by `breaks` are of equal length and that `center` is one of the breaks. This means we allow for more or less intervals on one side of `center`.

This really only works well if `colors` is specified as `g2r.colors`, which returns a symmetrical color vector (from green to red) if an even number of colors is requested. The whole point is then that if there are more classes to one side of `center` than to the other, this will be reflected by deeper shades of red or green on the appropriate side.

Value

A vector of colors, of length one less than the number of breaks.

See Also

`g2r.colors`

Examples

```
## Fully symmetrical breaks
br1 = (-3) : 3
co1 = breakColors(br1, g2r.colors)
co1
doLegend(br1, co1, 1)

## Truncated on one side
br2 = (-2) : 4
co2 = breakColors(br2, g2r.colors)
co2
doLegend(br2, co2, 1)

## Does not work with other color schemes
co3 = breakColors(br2, heat.colors)
co3
doLegend(br2, co3, 1)
```

`BrewerClusterCol` *Color scheme for clusters*

Description

This function returns a color vector based on one of the qualitative paletters supported by `RColorBrewer`. This allows visually distinct coloring of clusters and ensures sure that adjacent clusters have different colors.

Usage

```
BrewerClusterCol(n, name = "Pastel1")
```

Arguments

`n`	desired number of colors
`name`	name of the qualitative palette from which colors are taken, see `brewer.pal.info`

Details

This is just a wrapper for `brewer.pal` that checks that the specified palette is qualitative, and allows for an arbitrary number of colors: for less than three colors, it just returns the first and second colors of the palette; for more than `maxcolors` colors, it recycles the basic palette as often as required. This is ok, because the main point is to have different colors for neighboring clusters.

Value

A character vector of length `n` of hexadecimal color codes.

See Also

`brewer.pal`

Examples

```
## A Color Wheel: default palette with maximum number of colors
pie(rep(1,9), col=BrewerClusterCol(9))

## Double the number of colors
pie(rep(1,18), col=BrewerClusterCol(18))

## Only two clusters/colors
pie(rep(1,2), col=BrewerClusterCol(2))

## Different qualitative palette: stronger colors
pie(rep(1,12), col=BrewerClusterCol(12, "Paired"))
```

convAnnData *Converting data frames for display as annotation*

Description

Converts a data frames for display as annotation in a heatmap. This is mostly intended as an internal function, but might be useful for finetuning an annotation data frame manually.

Usage

```
convAnnData(x, nval.fac = 3, inclRef = TRUE, asIs = FALSE)
```

Arguments

x	the data frame to be converted
nval.fac	lower limit for unique values in numerical variables
inclRef	logical value indicating whether to include the reference level among the dummy variables for factors
asIs	logical value indicating whether to perform a conversion; if TRUE, the input x is simply returned, provided it is a numerical matrix (otherwise, the function stops with an error message)

Details

Logical variables are converted to factors. So are numerical variables with less than nval.fac unique values.

Value

convAnnData returns the converted data frame, which is a numerical matrix

See Also

annHeatmap2

Examples

```
data(mtcars)
summary(mtcars)
summary(convAnnData(mtcars))
summary(convAnnData(mtcars, nval.fac=2))
summary(convAnnData(mtcars, nval.fac=2, inclRef=FALSE))
```

doLegend *A simple legend*

Description

Add a simple legend in form of a color bar to a plot.

Usage

```
doLegend(breaks, col, side)
```

Arguments

breaks	a vector of breaks defining a set of intervals for the data
col	a vector of colors corresponding to the intervals.
side	integer between 1 and 4, indicating on which side of the main plot the legend is supposed to be drawn. Standard interpretation: 1 = below, continuing clockwise.

Details

This is an extremely simple way of giving a visual impression of what numerical values correspond to a given color. The actual plot is done via a call to `image` and `axis`.

Value

The locations of the ticks returned by the call to `axis`

See Also

`plot.annHeatmap`, `niceBreaks`, `g2r.colors`

Examples

```
## Set up data
doLegend(1:9, g2r.colors(8), 2)
```

g2r.colors *Palette from green to red via black*

Description

Returns a color vector of the requested length, ranging from pure red to pure green via slighlty tinted black.

Usage

```
g2r.colors(n = 12, min.tinge = 0.33)
```

Arguments

n	the number of requested colors
min.tinge	the proportion of red/green added to black to make it recognizably green or red

Details

If n is even, the colors range from pure green to green-tinted black to red-tinted black to pure red. If n is odd, the colors range from pure red to pure green, with full black for the median class.

Value

A vector of (RGB-) colors of the specified length

See Also

breakColors

Examples

```
## Even number: residual tint shows left/right of center
co_even = g2r.colors(10)
co_even
doLegend(1:11, co_even, 1)

## Odd number: central class all black
co_odd = g2r.colors(9)
co_odd
doLegend(1:10, co_odd, 1)

## Lighter tint in the middle
co_light = g2r.colors(10, min.tinge=0.50)
co_light
doLegend(1:11, co_light, 1)
```

heatmap_2 *Display Data as Heatmap*

Description

This function displays an expression data matrix as a heatmap. It is based on an old version of heatmap in the stats package, but offers more flexibility (e.g. skipping dendrograms, skipping row/column labelling, adding a legend).

Usage

```
heatmap_2(
  x,
  Rowv,
  Colv,
  distfun = dist,
  hclustfun = hclust,
  add.expr,
  scale = c("row", "column", "none"),
  na.rm = TRUE,
  do.dendro = c(TRUE, TRUE),
  legend = 0,
  legfrac = 8,
  col = heat.colors(12),
  trim,
  ...
)
```

Arguments

x	the numerical data matrix to be displayed.
Rowv	either a dendrogram or a vector of reordering indexes for the rows.
Colv	either a dendrogram or a vector of reordering indexes for the columns.
distfun	function to compute the distances between rows and columns. Defaults to dist.
hclustfun	function used to cluster rows and columns. Defaults to hclust.
add.expr	Expression to be evaluated after the call to image. See Details.
scale	indicates whether values should be scaled by either by row, column, or not at all. Defaults to row.
na.rm	logical indicating whther to remove NAs.
do.dendro	logical vector of length two, indicating (in this order) whether to draw the row and column dendrograms.
legend	integer between 1 and 4, indicating on which side of the plot the legend should be drawn, as in mtext.

legfrac	fraction of the plot that is taken up by the legend; larger values correspond to smaller legends.
col	the color scheme for image. The default sucks.
trim	Percentage of values to be trimmed. This helps to keep an informative color scale, see Details.
...	extra arguments to image.

Details

This function is deprecated. Please use regHeatmap for new projects.

With all parameters at their default, this gives the same result as a very old version of heatmap that was the base for the modifications. All parameters of the same name have the same function as in heatmap, though add.expr, which can be used for adding graphical elements after the call to image, will probably not produce useful results. Note also that row- and column labels are optional, i.e. if the corresponding dimname of x is NULL, no labels are displayed.

Note that setting Rowv or Colv to NA completely suppresses re-ordering of rows or columns as well as the corresponding dendrogram. Setting both to NA works basically like image (though you can still add a legend).

Setting trim to a number between 0 and 1 uses equidistant classes between the (trim)- and (1-trim)-quantile, and lumps the values below and above this range into separate open-ended classes. If the data comes from a heavy-tailed distribution, this can save the display from putting too many values into to few classes.

Value

Same as heatmap with keep.dendro=FALSE: an invisible list giving the reordered indices of the row- and column-elements as elements rowInd and colInd.

Author(s)

Original by Andy Liaw, with revisions by Robert Gentleman and Martin Maechler.

Alexander Ploner for this version.

See Also

heatmap, hclust, heatmap_plus, regHeatmap, annHeatmap

Examples

```
## Not run:
# create data
mm = matrix(rnorm(1000, m=1), 100,10)
mm = cbind(mm, matrix(rnorm(2000), 100, 20))
mm = cbind(mm, matrix(rnorm(1500, m=-1), 100, 15))
mm2 = matrix(rnorm(450), 30, 15)
mm2 = cbind(mm2, matrix(rnorm(900,m=1.5), 30,30))
mm=rbind(mm, mm2)
colnames(mm) = paste("Sample", 1:45)
```

```
rownames(mm) = paste("Gene", 1:130)

# similar to base heatmap
heatmap_2(mm)

# remove column dendrogram
heatmap_2(mm, do.dendro=c(TRUE, FALSE))

# add a legend under the plot
heatmap_2(mm, legend=1)
# make it smaller
heatmap_2(mm, legend=1, legfrac=10)
# ... on the left side
heatmap_2(mm, legend=2, legfrac=10)

# remove the column labels by removing the column names
colnames(mm)=NULL
heatmap_2(mm, legend=1, legfrac=10)

# truncate the data drastically
heatmap_2(mm, legend=1, legfrac=10, trim=0.1)

## End(Not run) ## end dontrun
```

---

heatmap_plus *Display an Annotated Heatmap*

Description

This function displays an expression data matrix as a heatmap with a column dendrogram. A given clustering will be shown in color. Additionally, a number of binary and interval scaled covariates can be added to characterize these clusters.

Usage

```
heatmap_plus(
  x,
  addvar,
  covariate = NULL,
  picket.control = list(),
  h,
  clus,
  cluscol,
  cluslabel = NULL,
  Rowv,
  Colv,
  reorder = c(TRUE, TRUE),
  distfun = dist,
```

```
  hclustfun = hclust,
  scale = c("row", "column", "none"),
  na.rm = TRUE,
  do.dendro = TRUE,
  col = heat.colors(12),
  trim,
  equalize = FALSE,
  ...
)
```

Arguments

x	the numerical data matrix to be displayed.
addvar	data frame with (mostly binary) covariates.
covariate	integer indicating the one column in addvar that is interval scaled.
picket.control	list of option for drawing the covariates, passed to oldPicketplot.
h	height at which to cut the dendrogram, as in oldCutree; overrides clus.
clus	an explicit vector of cluster memberships for the columns of x, if no dendrogram is used; ignored if do.dendro=TRUE and h is specified.
cluscol	a vector of colors used to indicate clusters.
cluslabel	labels to designate cluster names.
Rowv	either a dendrogram or a vector of reordering indexes for the rows.
Colv	either a dendrogram or a vector of reordering indexes for the columns.
reorder	logical vector of length two, indicating whether the rows and columns (in this order) should be reordered using order.dendrogram.
distfun	function to compute the distances between rows and columns. Defaults to dist.
hclustfun	function used to cluster rows and columns. Defaults to hclust.
scale	indicates whether values should be scaled by either by row, column, or not at all. Defaults to row.
na.rm	logical indicating whther to remove NAs.
do.dendro	logical indicating whether to draw the column dendrogram.
col	the color scheme for image. The default sucks.
trim	Percentage of values to be trimmed. This helps to keep an informative color scale, see Details.
equalize	logical indicating whther to use the ranks of the data for setting the color scheme; alternative to trim, see Details.
...	extra arguments to image.

Details

This function is deprecated. Please use functions annHeatmap or annHeatmap2 for new projects.

This is a heavily modified version of heatmap_2, which is a heavily modfied version of an old version of heatmap in package stats, so some of the arguments are described in more detail there.

The main distinguishing feature of this routine is the possibility to color a cluster solution, and to add a covariate display.

Covariates are assumed to be binary, coded as 0 and 1 (or FALSE and TRUE respectively). One of the covariates can be interval scaled, the column index of this variable is supplied via argument covariate. The details of the added display are handled by the function picketplot.

Setting trim to a number between 0 and 1 uses equidistant classes between the (trim)- and (1-trim)-quantile, and lumps the values below and above this range into separate open-ended classes. If the data comes from a heavy-tailed distribution, this can save the display from putting too many values into to few classes. Alternatively, you can set equal=TRUE, which uses an equidistant color scheme for the ranks of the values.

Value

A list with components

rowInd	indices of the rows of the display in terms of the rows of x.
colInd	ditto for the columns of the display.
clus	the cluster indices of the columns of the display.

Author(s)

Original by Andy Liaw, with revisions by Robert Gentleman and Martin Maechler.

Alexander Ploner for the modifications documented here.

See Also

heatmap_2, heatmap, oldPicketplot, oldCutplot.dendrogram, RGBColVec, annHeatmap, annHeatmap2

Examples

```
## Not run:
# create data
mm = matrix(rnorm(1000, m=1), 100,10)
mm = cbind(mm, matrix(rnorm(2000), 100, 20))
mm = cbind(mm, matrix(rnorm(1500, m=-1), 100, 15))
mm2 = matrix(rnorm(450), 30, 15)
mm2 = cbind(mm2, matrix(rnorm(900,m=1.5), 30,30))
mm=rbind(mm, mm2)
colnames(mm) = paste("Sample", 1:45)
rownames(mm) = paste("Gene", 1:130)
addvar = data.frame(Var1=rep(c(0,1,0),c(10,20,15)),
                    Var2=rep(c(1,0,0),c(10,20,15)),
                    Var3=rep(c(1,0), c(15,30)),
                    Var4=rep(seq(0,1,length=4), c(10,5,15,15))+rnorm(45, sd=0.5))
addvar[3,3] = addvar[17,2] = addvar[34,1] =NA
colnames(addvar) = c("Variable X","Variable Y", "ZZ","Interval")


# the lame default, without clustering
# Labels do not look too hot that way
```

```
heatmap_plus(mm)

# without labels, but with cluster
dimnames(mm)=NULL
heatmap_plus(mm, h=40)

# add some covariates, with nice names
heatmap_plus(mm, addvar=addvar, cov=4)

# covariates and clustering
heatmap_plus(mm, addvar=addvar, cov=4, h=20, col=RGBColVec(64), equal=TRUE)

# Clustering without the dendrogram
cc = cutree(hclust(dist(t(mm))), k=5)
heatmap_plus(mm, addvar=addvar, cov=4, clus=cc, do.dendro=FALSE)

## End(Not run) ## end dontrun
```

---

heatmapLayout *Generate a layout for an (annotated) heatmap*

Description

Generate a layout for an (annotated) heatmap. This function will generally not be called directly, but only via annHeatmap2.

Usage

```
heatmapLayout(dendrogram, annotation, leg.side = NULL, show = FALSE)
```

Arguments

dendrogram	A list with named entries Row and Col. Each of these is a list with a named entry status. If the value of status is the string "yes", space will be set aside for drawing a row- and/or column dendrogram.
annotation	A list with named entries Row and Col. Each of these is a list with a named entry data. If the value of data is not NULL, space will be set aside for a picket plot showing the row- and/or column annotation.
leg.side	An integer indicating on where to reserve space for the legend: values 1-4 correspond to below, to the left, above and to the right, as in e.g. axis. For a value of NULL, the function provides a reasonable default where there is space left in the layout. For any other value, no space for a legend is put aside.
show	A logical value; if TRUE, the layout defined by the arguments is displayed graphically.

Details

Space for plots is reserved via the `layout` mechanism. The function starts with an empty maximum layout, fills in the plot, dendrograms, annotation plots and legend as required, and compresses the resulting layout by removing empty slots.

Value

A list with the following entries:

`plot`	A matrix describing the plot layout; see `layout`
`width`	relative widths of plots (i.e. columns)
`height`	relative heights of plots (i.e. rows)
`legend.side`	side where to draw the legend

See Also

`annHeatmap2`, `picketPlot`, `layout`

Examples

```
def.par = par(no.readonly = TRUE) # save default, for resetting

## Heatmap with column dendrogram, column annotation, default legend
dnd = list(Row=list(status="no"), Col=list(status="yes"))
ann = list(Row=list(data=NULL), Col=list(data=1))
## 1 = heatmap, 2=dendrogram, 3=annotation, 4=legend
ll = heatmapLayout(dendrogram=dnd, annotation=ann, leg.side=NULL, show=TRUE)
ll

par(def.par)  #- reset to default
```

---

modifyExistingList *Override existing list entries*

Description

Override existing list entries and extract arguments that are specified as named lists

Usage

```
modifyExistingList(x, val)

extractArg(arglist, deflist)
```

Arguments

x	a named list, the target for replacing with entries with the same name from val
val	a named list that serves as template for filling in values in x
arglist	a named list; these are the specified arguments that override the defaults.
deflist	a named list whose entries are all possible slots (with default values) that can be filled.

Details

modifyExistingList is a general function that recursively overwrites named items in x with the value of items of val with the same name. Items in val that have no name, or do not correspond to an item in x with the same name, are ignored.

extractArg is a specific helper function for setting default values for the annHeatmap2-family of functions, where arguments are given as a list with two named items, Row and Col. Each of these items is again a named list of actual parameters. At the same time, all items with other names than Row and Col at the top level are assumed to be shared items with the same value for both sub-lists. extractArg uses modifyExistingList to overwrite the default values specified in deflist with the actual values specified in arglist, see Examples.

Value

modifyExistingList returns x, with values replaced from val where names match. extractArg returns a list with items Row and Col fully specified according to both deflist and arglist.

See Also

annHeatmap2

Examples

```
## Replace items with matching names recursively
x   = list(a=1, b=2, c=list(a=31, b=32), 135)
val = list(a=2, c=list(b=1114), d=92)
modifyExistingList(x, val)

## Same defaults for rows/columns, no arguments specified
defs = list(a="A", b="B", c="C")
extractArg(NULL, defs)

## Shared and non-shared defaults
defs = list(common.1=134, common.2=72, Row=list(row.only=14), Col=list(col.only=134))
args = list(common.1 = -1, Row=list(row.only=94, common.2=-15))
extractArg(args, defs)
```

`niceBreaks` *Get nice (symmetric) breaks for an interval*

Description

Given a minimum and a maximum, this function returns a vector of equidistant breaks that covers this interval, and has a pretty interval length (1, 2, or 5 times a power of 10). If the interval contains zero, it will be one of the breaks, so that the intervals are arranged somewhat symmetrically around it.

Usage

```
niceBreaks(xr, breaks)
```

Arguments

`xr`	the range to be covered, as `c(min, max)`
`breaks`	either the desired number of breaks, or a pre-specified vector of breaks

Details

The number of desired breaks is honored as far as possible, which is not actually that often in practice. However, major deviations of three or more are reasonably rare.

The functiona allows the specification of a set of breaks instead of the desired number of breaks, somewhat like in `cut`. However, if the length of `breaks` is greater than one, the function just sorts the values and returns them otherwise unchanged.

Value

A vector of pretty breaks covering the specified interval, more or less of the desired length.

See Also

`pretty`

Examples

```
## Niceness overrules specified number
niceBreaks(c(-1,1), 5)
niceBreaks(c(-1,1), 6)

## Zero appears always as break
niceBreaks(c(-2.75, 1.12), 8)

## Not invariant to translation (of course)
niceBreaks(3.27 + c(-2.75, 1.12), 8)
```

`oldCutplot.dendrogram` *Plot Subtrees of a Dendrogram in Different Colors*

Description

Plot a dendrogram, cut the tree at a given height, and draw the resulting subtrees in different colors (OLD version, to be deprecated)

Usage

```
oldCutplot.dendrogram(
  x,
  h,
  cluscol,
  leaflab = "none",
  horiz = FALSE,
  lwd = 3,
  ...
)
```

Arguments

`x`	a dendrogram.
`h`	the height at which the dendrogram is cut.
`cluscol`	the colors used for the subtrees; defaults to `rainbow`.
`leaflab`	indicates how leaf labels are to be drawn< defaults to 'perpendicular'.
`horiz`	logical indicating whether to plot the dendrogram horizontally or vertically.
`lwd`	the line width used for the color subtrees.
`...`	arguments to `plot.dendrogram`.

Details

This routine makes use of the functions `plot.dendrogram` and `plotNode` in package `stats`.

Author(s)

Alexander Ploner <Alexander.Ploner@ki.se>

See Also

`as.dendrogram`

Examples

```
## Not run:
data(swiss)
cc = as.dendrogram(hclust(dist(swiss)))
oldCutplot.dendrogram(cc, h=80)

## End(Not run)
```

`oldPicketplot` *Barplots for Several Binary Variables*

Description

Display one or more binary variables by using black bars for presence/validity of a condition, empty space for absence/invalidity, and an extra color for missing values. Additionally, an index plot for one interval scaled variable can be added, possibly with a smoothing function (OLD version, to be deprecated).

Usage

```
oldPicketplot(
  x,
  covariate = NULL,
  grp = NULL,
  grpcol,
  grplabel = NULL,
  add = FALSE,
  control = list()
)
```

Arguments

`x`	a matrix or data frame containing the data.
`covariate`	the index of the column in `x` that contains the interval scaled variable, if any.
`grp`	cluster indices for the rows of `x`, used for assigning background color.
`grpcol`	colors corresponding to the clusters.
`grplabel`	cluster names.
`add`	logical indicating whether to start a new plot, or whether to add the plot to the existing one.
`control`	a list of parameters controlling the appearance of the plot, see Details.

Details

This routine is primarily intended for augmenting heatmaps. It might be useful in other contexts, but misses most frills for using it comfortably.

The following named list elements can be set to change the appearance of the plot:

boxw the relative width of a marking box.

boxh the relative height of a marking box.

hbuff the horizontal separation around marking boxes; equals half the horizontal distance between two marking boxes.

vbuff ditto for vertical separation.

span passed on to `loess` used for the smoothing curve.

nacol color for missing values of binary variables.

degree if 0, no smoothing line is drawn; otherwise passed on to `loess` used for the smoothing curve.

cex.label the character size for `grplabel`.

Note

The plot looks like a more or less derelict picket fence, and 'picketplot' sounds somewhat like the 'pocketplot' used in geostatistics.

Author(s)

Alexander Ploner <Alexander.Ploner@ki.se>

See Also

heatmap_plus

Examples

```
## Not run:
# without covariate
mm = cbind(sample(0:1, 42, rep=TRUE), sample(0:1, 42, rep=TRUE))
mm[sample(42, 5), 1] = NA
oldPicketplot(mm)

# with clustering
cl = rep(1:3, c(10,22,10))
cn = c("Cluster I","Cluster II","Cluster III")
cc = c("lightblue","lightgreen","lightpink") # windows palette
oldPicketplot(mm, grp=cl, grplabel=cn, grpcol=cc)

# add a covariate; setting the colnames makes the variable labels
mm = cbind(mm, rnorm(42) + cl/2)
colnames(mm) = c("State A","State B", "X")
oldPicketplot(mm, covariate=3,grp=cl, grplabel=cn, grpcol=cc)
```

```
# using extra controls
oldPicketplot(mm, covariate=3,grp=cl, grplabel=cn, grpcol=cc, control=list(nacol="white", degree=0))

## End(Not run) ## end dontrun
```

`picketPlot` *Display a data frame of annotation information*

Description

Displays a data frame of both factor and numerical variables in parallel panels. Factors levels are indicated by black rectangles, using dummy variables for more than two levels. Numerical variables are shown as simple index plots with an optional loess smoother. Panels can be arranged horizontally or vertically, and different groups of subjects can be indicated through different background colors.

Usage

```
picketPlot(
  x,
  grp = NULL,
  grpcol,
  grplabel = NULL,
  horizontal = TRUE,
  asIs = FALSE,
  control = list()
)
```

Arguments

`x`	usually a data frame, which is passed to `convAnnData` to be converted to a numerical matrix with dummy coding for the factor levels. Alternatively, such a numerical matrix can be constructed manually and passed in as `x`, see argument `asIs`
`grp`	an optional vector of cluster memberships, in the same order as the rows of `x`
`grpcol`	an optional vector of background colors for the clusters specified in `grp`
`grplabel`	an optional vector of names for the clusters specified in `grp`
`horizontal`	logical value whether to plot variables horizontally (default) or vertically
`asIs`	a logical value indicating whether `x` should be passed to `convAnnData` for preprocessing or not. Defaults to `FALSE`.
`control`	a named list of control parameters that determines the visual appearance of the plot; see `picketPlotControl` for details.

Details

Missing values are indicated by a box marking in `nacol` for factor values.

Value

Invisibly, a list containing the data and parameters used for plotting each binary indicator and numerical variable, respectively. This is an internal data structure, mostly useful for debugging. Irrelevant, as the main desired effetc is a plot to the current graphical device.

See Also

annHeatmap2, convAnnData, par, picketPlotControl

Examples

```
## Standard call
data(mtcars)
picketPlot(mtcars)

## Pre-process the data for display
mm = convAnnData(mtcars, inclRef=FALSE)
picketPlot(mm, asIs=TRUE)

## Higher panels for continous traits
picketPlot(mm, asIs=TRUE, control=list(numfac=3))

## With clusters
picketPlot(mtcars, grp = rep(1:2, c(16, 16)), grpcol = c("pink","lightblue"), grplabel=c("Cluster 1", "Cluster 2
```

---

picketPlotControl *Default parameter settings for picketPlot*

Description

This function returns a named list of parameters that affect how a picketPlot is generated. This list can be used as a template for overriding the defaults partially or completely.

Usage

```
picketPlotControl()
```

Details

The following parameter affects the overall appearance of the plot:

- `cex.label` is the expansion factor for the size of the cluster labels at the bottom of the plot; default is 1.5.

The following parameters directly affect how binary indicator variables are displayed:

- `boxw` is the relative length of the short side of a box marking (width for a horizontal plot); default is 1.

- boxh is the relative length of the long side of a box marking (default: 4)
- hbuff is the relative distance between two box markings for the same variable (horizontal buffer for a horizontal plot); default is 0.1
- vbuff is the relative distance between two box markings for the same subject, but different variables (default: 0.1)
- nacol is the color for box markings indicating missing values (default: gray(0.85))

Note that boxh and vbuff also affect the display of numerical variables as a scatter plot: as the amount of vertical space allowed for a single numerical variable (see also numfac below) and the vertical space between two neighboring variable panels (binary or ornumerical), respectively.

The following parameters only affect the display of a numerical variable:

- numfac is the expansion factor indicating how much higher (for a horizontal plot) or wider (for a vertical plot) panels with numerical variables are than a panels for a single binary indicator
- span is the span argument for the loess smoother. Default is 1/3; setting this to zero switches off smoothing.
- degree is the degree of loess smoothing. Default is 1; setting this to zero switches off smoothing
- pch is the plotting character for numerical variables; uses the device default.
- cex.pch is the size of the plotting character for numerical variables; uses the device default.
- col.pch is the color of the plotting character for numerical variables; uses the device default.
- label_axis_shrink controls the range of the axis for which axis ticks are labeled: by default, labels covering the whole observed range are defined (via a call to pretty); if set to a number between zero and one, the range covered by labels is shortened by that fraction and centered within the observed range (and the fed to pretty); this can be used to avoid overlapping labels for multiple adjacent panes with numerical variables.
- plot_baseline is a logical value indicating whether to draw a baseline for panes showing numerical variables: FALSE by default, this can be useful to visually separate multiple adjacent panes with numerical variables.

Value

A named list

See Also

picketPlot, par, pretty

`plot.annHeatmap` *Plotting method for annotated heatmaps*

Description

Plotting method for annotated heatmaps

Usage

```
## S3 method for class 'annHeatmap'
plot(x, widths, heights, ...)
```

Arguments

`x`	an object of class `annHeatmap`
`widths`	a numerical vector giving the widths of the sub-plots currently defined
`heights`	a numerical vector giving the heights of the sub-plots currently defined
`...`	extra graphical parameters, currently ignored

Details

This function displays an annotated heatmap object that has been previously generated by `annHeatmap2` or on of its wrappers. The arguments `widths` and `heights` work as in `layout`.

Value

`x`, invisibly returned. If `widths` or `heights` have been specified, they overwrite the corresponding items `x$layout$width` and `x$layout$height` in `x`.

See Also

`annHeatmap2`, `heatmapLayout`, `layout`

Examples

```
## Define the map
require(Biobase)
data(sample.ExpressionSet)
ex1 = sample.ExpressionSet[51:85,]
map1 = annHeatmap2(exprs(ex1), ann=list(Col=list(data=pData(ex1))),
               cluster=list(Col=list(cuth=3000)))

## Plot it
plot(map1)

## More heatmap, smaller dendrogram/annotation
map2 = plot(map1, heights = c(1,6,1))
```

```
## Compare layout before/after
with(map1$layout, layout(plot, width, height))
layout.show(4)
with(map2$layout, layout(plot, width, height))
layout.show(4)
```

`print.annHeatmap` *Printing information about annotated heatmaps*

Description

Printing method for annotated heatmaps

Usage

```
## S3 method for class 'annHeatmap'
print(x, ...)
```

Arguments

`x`	an object of class `annHeatmap`
`...`	extra arguments, currently ignored

Details

A very simple printing method, displaying a minimum of information about dendrograms and annotation

Value

`x` is returned invisibly

See Also

`annHeatmap`, `annHeatmap2`, `plot.annHeatmap`

Examples

```
set.seed(219)
mat = matrix(rnorm(100), ncol=5)
ann = data.frame(Class=c("A","A","B","A","B"))
map1 = annHeatmap(mat, ann)
map1
```

regHeatmap *Regular heatmaps with a legend*

Description

Creating regular heatmaps, without annotation, but allowing for a legend

Usage

```
regHeatmap(x, ...)

## Default S3 method:
regHeatmap(
  x,
  dendrogram = list(clustfun = hclust, distfun = dist, status = "yes"),
  labels = NULL,
  legend = TRUE,
  ...
)
```

Arguments

x	a numerical matrix
...	extra options passed to annHeatmap2
dendrogram	a list controlling the options for row- and column dendrogram, see annHeatmap2
labels	a list controlling the row- and column labels as well as their location and size, see annHeatmap2
legend	either a logical value, indicating whether to draw a legend at the default location determined by the function, or one of the sides of the plot (1-4), see annHeatmap2

Details

A gelded wrapper for annHeatmap2 that allows for heatmaps without annotation or clustering on the dendrograms, but still offer some control over dendrograms, labels and legend.

These functions generate an object representing the heatmap; in order to produce graphical output, you have to invoke the plot method, see Examples.

Value

An object of class annHeatmap

See Also

annHeatmap, annHeatmap2, plot.annHeatmap

Examples

```
## Default
set.seed(219)
mat = matrix(rnorm(100), ncol=5)
map1 = regHeatmap(mat)
plot(map1)
```

RGBColVec *Alternative color schemes*

Description

RGBColVec returns a vector of colors that is equally spaced from red through black to green, suitable for heatmaps.

Usage

```
RGBColVec(nrgcols = 12)

RainbowPastel(n, blanche = 200, ...)
```

Arguments

nrgcols, n	desired number of colors
blanche	the amount of whiteness added; value between 0 and 255
...	extra arguments to rainbow

Details

RainbowPastel returns a vector of colors like rainbow, but more pastelly.

Value

A character vector of length nrgcols or n giving the RGB codes for the colors.

Author(s)

RGBColVec is based on function rgcolors.func in package sma by Sandrine Dudoit and Jane Fridlyand.

RGBColVec as documented and RainbowPastel by Alexander Ploner

See Also

heat.colors

Examples

```
## Not run:
 # A Color Wheel
pie(rep(1,12), col=RGBColVec(12))

# A color wheel in the original rainbow
pie(rep(1,6), col=rainbow(6))

# Pastel
pie(rep(1,6), col=RainbowPastel(6))

# Less whiteness
pie(rep(1,6), col=RainbowPastel(6, blanche=127))

# More steps require less whiteness
pie(rep(1,12), col=RainbowPastel(12, blanche=60))

# Test your screen & eyes: any differences?
pie(rep(1,12), col=RainbowPastel(12, blanche=80))

## End(Not run) ## end dontrun
```

Index

∗ **aplot**
  oldCutplot.dendrogram, 21
  oldPicketplot, 22
∗ **cluster**
  oldCutplot.dendrogram, 21
∗ **color**
  BrewerClusterCol, 8
  RGBColVec, 30
∗ **hplot**
  annHeatmap, 2
  annHeatmap2, 4
  heatmap_2, 12
  heatmap_plus, 14
  oldCutplot.dendrogram, 21
  oldPicketplot, 22
  picketPlot, 24
  plot.annHeatmap, 27
  print.annHeatmap, 28
  regHeatmap, 29
∗ **utilities**
  breakColors, 6
  convAnnData, 9
  doLegend, 10
  g2r.colors, 11
  heatmapLayout, 17
  modifyExistingList, 18
  niceBreaks, 20

annHeatmap, 2, *13*, *16*, *28, 29*
annHeatmap2, *3*, 4, *9*, *16*, *18, 19*, *25*, *27–29*
as.dendrogram, *21*
axis, *10*

breakColors, 6, *6*, *11*
brewer.pal, *8*
brewer.pal.info, *8*
BrewerClusterCol, *6*, 8

convAnnData, 9, *25*

doLegend, 10

extractArg *(*modifyExistingList*)*, 18

g2r.colors, *6, 7*, *10*, 11

hclust, *13*
heat.colors, *30*
heatmap, *13*, *16*
heatmap_2, 12, *16*
heatmap_plus, *13*, 14, *23*
heatmapLayout, *6*, 17, *27*

image, *10*

layout, *18*, *27*

modifyExistingList, 18

niceBreaks, *6*, *10*, 20

oldCutplot.dendrogram, *16*, 21
oldPicketplot, *16*, 22

par, *25, 26*
picketPlot, *18*, 24, *26*
picketPlotControl, 25, *25*
plot.annHeatmap, *3*, *10*, 27, *28, 29*
pretty, *20*, *26*
print.annHeatmap, 28

RainbowPastel *(*RGBColVec*)*, 30
regHeatmap, *13*, 29
RGBColVec, *16*, 30