Package 'heatmaply'

May 8, 2026

Type Package

Title Interactive Cluster Heat Maps Using 'plotly' and 'ggplot2'

Version 1.6.0

Date 2025-07-12

Description Create interactive cluster 'heatmaps' that can be saved as a stand-alone HTML file, embedded in 'R Markdown' documents or in a 'Shiny' app, and available in the 'RStudio' viewer pane. Hover the mouse pointer over a cell to show details or drag a rectangle to zoom. A 'heatmap' is a popular graphical method for visualizing high-dimensional data, in which a table of numbers are encoded as a grid of colored cells. The rows and columns of the matrix are ordered to highlight patterns and are often accompanied by 'dendrograms'. 'Heatmaps' are used in many fields for visualizing observations, correlations, missing values patterns, and more. Interactive 'heatmaps' allow the inspection of specific value by hovering the mouse over a cell, as well as zooming into a region of the 'heatmap' by dragging a rectangle around the relevant area. This work is based on the 'ggplot2' and 'plotly.js' engine. It produces similar 'heatmaps' to 'heatmap.2' with the advantage of speed ('plotly.js' is able to handle larger size matrix), the ability to zoom from the 'dendrogram' panes, and the placing of factor variables in the sides of the 'heatmap'.

Depends R (>= 3.0.0), plotly (>= 4.7.1), viridis

Imports ggplot2 (>= 2.2.0), dendextend (>= 1.12.0), magrittr (>= 1.0.1), reshape2, scales, seriation, utils, stats, grDevices, methods, colorspace, RColorBrewer, htmlwidgets, webshot, assertthat, egg

Suggests knitr, covr, gplots, tidyselect, rmarkdown, testthat

VignetteBuilder knitr

License GPL-2 | GPL-3

URL https://talgalili.github.io/heatmaply/, https://cran.r-project.org/package=heatmaply, https://github.com/talgalili/heatmaply/, https://www.r-statistics.com/tag/heatmaply/

BugReports https://github.com/talgalili/heatmaply/issues

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Tal Galili [aut, cre, cph] (https://www.r-statistics.com),
Alan O'Callaghan [aut] (https://github.com/Alanocallaghan),
Jonathan Sidi [ctb] (https://github.com/yonicd),
Jaehyun Joo [ctb] (https://github.com/jaehyunjoo),
Yoav Benjamini [ths],
Mathew Simenc [ctb] (https://gitlab.com/mcsimenc,
https://github.com/mcsimenc)

Maintainer Tal Galili <tal.galili@gmail.com>

Repository CRAN

Date/Publication 2025-07-12 21:20:02 UTC

Contents

`ggheatmap` *ggplot heatmap equivalent to heatmaply*

Description

This function produces a ggplot analogue of heatmaply figures using ggarrange. This function may not always support the same set of features as , and exporting the heatmaply object with, for example, orca or `heatmaply(mtcars, file = "foo.png")`.

Usage

```
ggheatmap(
  ...,
  widths = NULL,
  heights = NULL,
  row_dend_left = FALSE,
  hide_colorbar = FALSE
)
```

Arguments

`...`	Passed to heatmaply
`widths, heights`	Relative widths and heights of plots.
`row_dend_left`	Logical argument controlling whether the row dendrogram is placed on the left of the plot.
`hide_colorbar`	Logical argument controlling whether the color bar (i.e.: the legend) is hidden.

Examples

```
ggheatmap(mtcars)
```

---

ggplot_side_color_plot

Side color plots for heatmaps

Description

Important for creating annotation.

Usage

```
ggplot_side_color_plot(
  df,
  palette = NULL,
  scale_title = paste(type, "side colors"),
  type = c("column", "row"),
  text_angle = if (type == "column") 0 else 90,
  is_colors = FALSE,
  fontsize = 10,
  label_name = NULL
)
```

Arguments

df	A "molten" data.frame as produced by (eg) reshape2::melt
palette	A function which can return colors to be used in the sidebar plot
scale_title	Title of the color scale. Not currently used.
type	Horizontal or vertical plot? Valid values are "column" and "row"
text_angle	the angle of the text of the rows/columns.
is_colors	Use if the values in df are valid colours and should not be mapped to a color scheme, and instead should be plotted directly.
fontsize	Font size (currently unused)
label_name	Name for the mouseover label, usually "row" or "column"

Value

A ggplot geom_tile object

---

heatmaply *Cluster heatmap based on plotly*

Description

An object of class heatmapr includes all the needed information for producing a heatmap. The goal is to separate the pre-processing of the heatmap elements from the graphical rendering of the object, which could be done

(Please submit an issue on github if you have a feature that you wish to have added)

heatmaply_na is a wrapper for 'heatmaply' which comes with defaults that are better for exploring missing value (NA) patterns. Specifically, the grid_gap is set to 1, and the colors include two shades of grey. It also calculates the is.na10 automatically.

heatmaply_cor is a wrapper for 'heatmaply' which comes with defaults that are better for correlation matrixes. Specifically, the limits are set from -1 to 1, and the color palette is RdBu.

Usage

```
heatmaply(x, ...)

heatmaply_na(x, grid_gap = 1, colors = c("grey80", "grey20"), ...)

heatmaply_cor(x, limits = c(-1, 1), colors = cool_warm, ...)

## Default S3 method:
heatmaply(
  x,
  colors = viridis(n = 256, alpha = 1, begin = 0, end = 1, option = "viridis"),
  limits = NULL,
```

```
  na.value = "grey50",
  row_text_angle = 0,
  column_text_angle = 45,
  subplot_margin = 0,
  cellnote = NULL,
  draw_cellnote = !is.null(cellnote),
  cellnote_color = "auto",
  cellnote_textposition = "middle right",
  cellnote_size = 12,
  Rowv = NULL,
  Colv = NULL,
  distfun = stats::dist,
  hclustfun = stats::hclust,
  dist_method = NULL,
  hclust_method = NULL,
  distfun_row = distfun,
  hclustfun_row = hclustfun,
  distfun_col = distfun,
  hclustfun_col = hclustfun,
  dendrogram = c("both", "row", "column", "none"),
  show_dendrogram = c(TRUE, TRUE),
  reorderfun = function(d, w) reorder(d, w),
  k_row = 1,
  k_col = 1,
  symm = FALSE,
 revC = symm || (is.dendrogram(Colv) & is.dendrogram(Rowv) & identical(Rowv, rev(Colv))),
  scale = c("none", "row", "column"),
  na.rm = TRUE,
  row_dend_left = FALSE,
  margins = c(NA, NA, NA, NA),
  ...,
  scale_fill_gradient_fun = NULL,
  grid_color = NA,
  grid_gap = 0,
  srtRow = NULL,
  srtCol = NULL,
  xlab = "",
  ylab = "",
  main = "",
  titleX = TRUE,
  titleY = TRUE,
  hide_colorbar = FALSE,
  key.title = NULL,
  return_ppxpy = FALSE,
  row_side_colors = NULL,
  row_side_palette = NULL,
  col_side_colors = NULL,
  col_side_palette = NULL,
```

```
  ColSideColors = NULL,
  RowSideColors = NULL,
  seriate = c("OLO", "mean", "none", "GW"),
  heatmap_layers = NULL,
  side_color_layers = NULL,
  dendrogram_layers = NULL,
  branches_lwd = 0.6,
  file = NULL,
  width = NULL,
  height = NULL,
  long_data = NULL,
  plot_method = c("ggplot", "plotly"),
  label_names = c("row", "column", "value"),
  fontsize_row = 10,
  fontsize_col = 10,
  cexRow = NULL,
  cexCol = NULL,
  subplot_widths = NULL,
  subplot_heights = NULL,
  colorbar_len = 0.3,
  colorbar_thickness = 30,
  colorbar_xanchor = if (row_dend_left) "right" else "left",
  colorbar_yanchor = "bottom",
  colorbar_xpos = if (row_dend_left) -0.1 else 1.1,
  colorbar_ypos = 0,
  showticklabels = c(TRUE, TRUE),
  dynamicTicks = FALSE,
  grid_size = 0.1,
  node_type = "heatmap",
  point_size_mat = NULL,
  point_size_name = "Point size",
  label_format_fun = function(...) format(..., digits = 4),
  labRow = NULL,
  labCol = NULL,
  custom_hovertext = NULL,
  suppress_default_hovertext = FALSE,
  col = NULL,
  dend_hoverinfo = TRUE,
  side_color_colorbar_len = 0.3,
  plotly_source = "A"
)

## S3 method for class 'heatmapr'
heatmaply(
  x,
  colors = viridis(n = 256, alpha = 1, begin = 0, end = 1, option = "viridis"),
  limits = NULL,
  na.value = "grey50",
```

```
  row_text_angle = 0,
  column_text_angle = 45,
  subplot_margin = 0,
  row_dend_left = FALSE,
  margins = c(NA, NA, NA, NA),
  ...,
 scale_fill_gradient_fun = scale_fill_gradientn(colors = if (is.function(colors))
   colors(256) else colors, na.value = na.value, limits = limits),
  grid_color = NA,
  grid_gap = 0,
  srtRow = NULL,
  srtCol = NULL,
  xlab = "",
  ylab = "",
  main = "",
  titleX = TRUE,
  titleY = TRUE,
  hide_colorbar = FALSE,
  key.title = NULL,
  return_ppxpy = FALSE,
  draw_cellnote = FALSE,
  cellnote_color = "auto",
  cellnote_textposition = "middle right",
  cellnote_size = 12,
  row_side_colors = x[["row_side_colors"]],
  row_side_palette = NULL,
  col_side_colors = x[["col_side_colors"]],
  col_side_palette = NULL,
  plot_method = c("ggplot", "plotly"),
  ColSideColors = NULL,
  RowSideColors = NULL,
  heatmap_layers = NULL,
  side_color_layers = NULL,
  dendrogram_layers = NULL,
  branches_lwd = 0.6,
  label_names = c("row", "column", "value"),
  fontsize_row = 10,
  fontsize_col = 10,
  subplot_widths = NULL,
  subplot_heights = NULL,
  colorbar_xanchor = if (row_dend_left) "right" else "left",
  colorbar_yanchor = "bottom",
  colorbar_xpos = if (row_dend_left) -0.1 else 1.1,
  colorbar_ypos = 0,
  colorbar_len = 0.3,
  colorbar_thickness = 30,
  showticklabels = c(TRUE, TRUE),
  dynamicTicks = FALSE,
```

```
  node_type = c("scatter", "heatmap"),
  grid_size = 0.1,
  point_size_mat = x[["matrix"]][["point_size_mat"]],
  point_size_name = "Point size",
  label_format_fun = function(...) format(..., digits = 4),
  custom_hovertext = x[["matrix"]][["custom_hovertext"]],
  suppress_default_hovertext = FALSE,
  dend_hoverinfo = TRUE,
  side_color_colorbar_len = 0.3,
  plotly_source = "A",
  height = NULL,
  width = NULL
)
```

Arguments

x	can either be a heatmapr object, or a numeric matrix Defaults to TRUE unless x contains any NAs.
...	other parameters passed to heatmapr (currently, various parameters may be ignored.
grid_gap	this is a fast alternative to grid_color. The default is 0, but if a larger value is used (for example, 1), then the resulting heatmap will have a white grid which can help identify different cells. This is implemented using style (with xgap and ygap).
colors, col	a vector of colors to use for heatmap color. The default uses viridis(n=256, alpha = 1, begin = 0, end = 1, option = "viridis") It is passed to scale_fill_gradientn. If colors is a color function (with the first argument being 'n' = the number of colors), it will be used to create 256 colors from that function. (col is there to stay compatible with heatmap.2)
limits	a two dimensional numeric vector specifying the data range for the scale.
na.value	color to use for missing values (default is "grey50").
row_text_angle	numeric (Default is 0), the angle of the text of the rows. (this is called srtRow in heatmap.2)
column_text_angle	numeric (Default is 45), the angle of the text of the columns. (this is called srtCol in heatmap.2)
subplot_margin	Currently not well implemented. It is passed to subplot. Default is 0. Either a single value or four values (all between 0 and 1). If four values are provided, the first is used as the left margin, the second is used as the right margin, the third is used as the top margin, and the fourth is used as the bottom margin. If a single value is provided, it will be used as all four margins.
cellnote	Values to be shown as annotations atop the heatmap cells.
draw_cellnote	Should the cellnote annotations be drawn? Defaults is FALSE, if cellnote is not supplied, TRUE if cellnote is supplied. If TRUE and cellnote is not supplied, x will be used for cellnote.
cellnote_color	The color of the cellnote text to be used.

`cellnote_textposition`
: The text positioning/centering of the cellnote. Default is "middle right". Options are "top left", "top center", "top right", "middle left", "middle center", "middle right", "bottom left", "bottom center", "bottom right"

`cellnote_size`
: The font size (HTML/CSS) of the cellnote. Default is 12.

`Rowv`
: determines if and how the row dendrogram should be reordered. By default, it is TRUE, which implies dendrogram is computed and reordered based on row means. If NULL or FALSE, then no dendrogram is computed and no reordering is done. If a dendrogram (or hclust), then it is used "as-is", ie without any reordering. If a vector of integers, then dendrogram is computed and reordered based on the order of the vector.

`Colv`
: determines if and how the column dendrogram should be reordered. Has the options as the Rowv argument above and additionally when x is a square matrix, Colv = "Rowv" means that columns should be treated identically to the rows.

`distfun`
: function used to compute the distance (dissimilarity) between both rows and columns. Defaults to dist. The options "pearson", "spearman" and "kendall" can be used to use correlation-based clustering, which uses `as.dist(1 - cor(t(x)))` as the distance metric (using the specified correlation method).

`hclustfun`
: function used to compute the hierarchical clustering when Rowv or Colv are not dendrograms. Defaults to hclust.

`dist_method`
: default is NULL (which results in "euclidean" to be used). Can accept alternative character strings indicating the method to be passed to distfun. By default distfun. is dist hence this can be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".

`hclust_method`
: default is NULL (which results in "complete" to be used). Can accept alternative character strings indicating the method to be passed to hclustfun By default hclustfun is hclust hence this can be one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC). Specifying hclust_method=NA causes heatmaply to use `find_dend` to find the "optimal" dendrogram for the data.

`distfun_row`
: distfun for row dendrogram only.

`hclustfun_row`
: hclustfun for col dendrogram only.

`distfun_col`
: distfun for row dendrogram only.

`hclustfun_col`
: hclustfun for col dendrogram only.

`dendrogram`
: character string indicating whether to compute 'none', 'row', 'column' or 'both' dendrograms. Defaults to 'both'. However, if Rowv (or Colv) is FALSE or NULL and dendrogram is 'both', then a warning is issued and Rowv (or Colv) arguments are honoured. It also accepts TRUE/FALSE as synonyms for "both"/"none".

`show_dendrogram`
: Logical vector of length two, controlling whether the row and/or column dendrograms are displayed. If a logical scalar is provided, it is repeated to become a logical vector of length two.

`reorderfun`
: function(d, w) of dendrogram and weights for reordering the row and column dendrograms. The default uses reorder.dendrogram

k_row	an integer scalar with the desired number of groups by which to color the dendrogram's branches in the rows (uses color_branches) If NA then find_k is used to deduce the optimal number of clusters.
k_col	an integer scalar with the desired number of groups by which to color the dendrogram's branches in the columns (uses color_branches) If NA then find_k is used to deduce the optimal number of clusters.
symm	logical indicating if x should be treated symmetrically; can only be true when x is a square matrix.
revC	logical indicating if the column order should be reversed for plotting. Default (when missing) - is FALSE, unless symm is TRUE. This is useful for cor matrix.
scale	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. The default is "none".
na.rm	logical (default is TRUE) indicating whether NA's should be removed when scaling (i.e.: when using rowMeans/colMeans). Generally it should always be kept as TRUE, and is included here mainly to stay backward compatible with gplots::heatmap.2. This argument does not effect the presence of NA values in the matrix itself. For removing rows/columns with NAs you should pre-process your matrix using na.omit (or some form of imputation).
row_dend_left	logical (default is FALSE). Should the row dendrogram be plotted on the left side of the heatmap. If false then it will be plotted on the right side.
margins	numeric vector of length 4 (default is c(50,50,NA,0)) containing the margins (see layout) for column, row and main title names, respectively. The top margin is NA by default. If main=="" then the top margin will be set to 0, otherwise it will get 30. For a multiline title a larger default for the 3rd element should be set. The right margin is NA by default, meaning it will be zero if row_dend_left is FALSE, or 100 if row_dend_left is TRUE.
scale_fill_gradient_fun	A function that creates a smooth gradient for the heatmap. The default uses scale_fill_gradientn with the values of colors, limits, and na.value that are supplied by the user. The user can input a customized function, such as scale_color_gradient() in order to get other results (although the virids default is quite recommended)
grid_color	control the color of the heatmap grid. Default is NA. Value passed to geom_tile. Do not use this parameter on larger matrix sizes, as it can dramatically prolong the build time of the heatmap. (another parameter, grid_color, will be added in the future - once it is implemented in plotly) In the meantime it is MUCH better to use the grid_gap argument.
srtRow	if supplied, this overrides row_text_angle (this is to stay compatible with heatmap.2)
srtCol	if supplied, this overrides column_text_angle (this is to stay compatible with heatmap.2)
xlab	A character title for the x axis.
ylab	A character title for the y axis.
main	A character title for the heatmap.
titleX	logical (TRUE). should x-axis titles be retained? (passed to subplot).
titleY	logical (TRUE). should y-axis titles be retained? (passed to subplot).

`hide_colorbar` logical (FALSE). If TRUE, then the color bar (i.e.: the legend) is hidden.

`key.title` (character) main title of the color key. If set to NULL (default) no title will be plotted.

`return_ppxpy` logical (FALSE). If TRUE, then no plotting is done and the p, px and py objects are returned (before turning into plotly objects). This is a temporary option which might be removed in the future just to make it easy to create a ggplot heatmaps.

`row_side_colors`, `col_side_colors`
data.frame of factors to produce row/column side colors in the style of heatmap.2/heatmap.3. When a data.frame is provided, the column names are used as the label names for each of the newly added row_side_colors. When a vector is provided it is coerced into a data.frame and the name of the side color will be just row_side_colors.

`row_side_palette`, `col_side_palette`
Color palette functions to be used for row_side_colors and col_side_colors respectively.

`ColSideColors`, `RowSideColors`
passed to row_side_colors,col_side_colors in order to keep compatibility with heatmap.2

`seriate` character indicating the method of matrix sorting (default: "OLO"). Implemented options include: "OLO" (Optimal leaf ordering, optimizes the Hamiltonian path length that is restricted by the dendrogram structure - works in O(n^4)) "mean" (sorts the matrix based on the reorderfun using marginal means of the matrix. This is the default used by heatmap.2), "none" (the default order produced by the dendrogram), "GW" (Gruvaeus and Wainer heuristic to optimize the Hamiltonian path length that is restricted by the dendrogram structure)

`heatmap_layers` ggplot object(s) (eg, list(theme_bw())) to be added to the heatmap before conversion to a plotly object.

`side_color_layers`
ggplot2 objects to be added to side color plots, similar to heatmap_layers.

`dendrogram_layers`
ggplot2 objects to be added to dendrograms, similar to heatmap_layers and side_color_layers.

`branches_lwd` numeric (default is 0.6). The width of the dendrograms' branches. If NULL then it is ignored. If the "lwd" is already defined in Rowv/Colv then this parameter is ignored (it is checked using has_edgePar("lwd")).

`file` name of the file(s) into which to save the heatmaply output. Should be a character vector of strings ending with ".html" for a dynamic output, or ".png", ".jpeg", ".pdf" for a static output.

For example: heatmaply(x, file = "heatmaply_plot.html") or dir.create("folder"); heatmaply(x, file = "folder/heatmaply_plot.html")

This is based on saveWidget, and webshot for the static files. For more refined control over the static file output, you should save the heatmaply object using export and pass the arguments you want based on the ones in webshot.

Another example: heatmaply(x, file = c("heatmaply_plot.html", "heatmaply_plot.png"))

`width`, `height` The width and height of the output htmlwidget, or the output file if exporting to png/pdf/etc. Presumed to be in pixels, but if a plotly internal function decides it's in other units you may end up with a huge file! Default is 800x500 when exporting to a file, and 100 as a htmlwidget.

`long_data` Data in long format. Replaces x, so both should not be used. Colnames must be c("name", "variable", "value"). If you do not have a names column you can simply use a sequence of numbers from 1 to the number of "rows" in the data.

`plot_method` Use "ggplot" or "plotly" to choose which library produces heatmap and dendrogram plots

`label_names` Names for labels of x, y and value/fill mouseover.

`fontsize_row`, `fontsize_col`, `cexRow`, `cexCol`
Font size for row and column labels.

`subplot_widths`, `subplot_heights`
The relative widths and heights of each subplot. The length of these vectors will vary depending on the number of plots involved.

`colorbar_len` The length of the colorbar/color key relative to the total plot height. Only used if plot_method = "plotly"

`colorbar_thickness`
The thickness (width) of the colorbar/color key in pixels. Only used if plot_method = "plotly".

`colorbar_xanchor`, `colorbar_yanchor`
The x and y anchoring points of the colorbar/color legend. Can be "left", "middle" or "right" for colorbar_xanchor, and "top", "middle" or "bottom" for colorbar_yanchor. See `colorbar` for more details.

`colorbar_xpos`, `colorbar_ypos`
The x and y co-ordinates (in proportion of the plot window) of the colorbar/color legend. See `colorbar` for more details.

`showticklabels` A logical vector of length two (default is TRUE). If FALSE, then the ticks are removed from the sides of the plot. The first location refers to the x axis and the second to the y axis. If only one value is supplied (TRUE/FALSE) then it is replicated to get to length 2. When using this parameter, it might be worth also adjusting margins. This option should be used when working with medium to large matrix size as it makes the heatmap much faster (and the hover still works).

`dynamicTicks` (default: FALSE). passed to ggplotly: should plotly.js dynamically generate axis tick labels? Dynamic ticks are useful for updating ticks in response to zoom/pan interactions; however, they can not always reproduce labels as they would appear in the static ggplot2 image.

`grid_size` When node_type is "scatter", this controls point size. When node_type is "heatmap", this controls the size of the grid between heatmap cells.

`node_type` For plot_method = "ggplot", should the heatmap be rendered as a x-y scatter plot (node_type = "scatter") or a heatmap (node_type = "heatmap"). Default is node_type = "heatmap".

`point_size_mat` Matrix to map to point size

`point_size_name`
Name of point size mapping (for hovertext/legend)

label_format_fun
: Function to format hovertext (eg, `function(...) round(..., digits=3)` or `function(...) format(..., digits=3)`

labRow, labCol
: character vectors with row and column labels to use; these default to rownames(x) or colnames(x), respectively. if set to NA, they change the value in showticklabels to be FALSE. This is mainly to keep backward compatibility with gplots::heatmap.2.

custom_hovertext
: Custom hovertext matrix (the same dimensions as the input). If plot_method is "plotly" then just this text is displayed; if plot_method if "ggplot" then it is appended to the existing text.

suppress_default_hovertext
: Logical indicating whether to hide the default hovertext for plot_method = "ggplot" of row, column, value, and Point size.

dend_hoverinfo
: Boolean value which controls whether mouseover text is shown for the row and column dendrograms.

side_color_colorbar_len
: As with colorbar_len, this controls the length of the colorbar/color key relative to the total plot height. This argument controls the colorbar_len of the side colour plots. Only used if plot_method = "plotly".

plotly_source
: See source argument in plot_ly

Examples

```
## Not run:

# mtcars
# x <- heatmapr(mtcars)
library(heatmaply)
heatmaply(iris[, -5], k_row = 3, k_col = 2)
heatmaply(cor(iris[, -5]))
heatmaply(cor(iris[, -5]), limits = c(-1, 1))
heatmaply(mtcars, k_row = 3, k_col = 2)
# heatmaply(mtcars, k_row = 3, k_col = 2, grid_color = "white")
heatmaply(mtcars, k_row = 3, k_col = 2, grid_gap = 1)

# make sure there is enough room for the labels:
heatmaply(mtcars, margins = c(40, 130))
# this is the same as using:
heatmaply(mtcars) %>% layout(margin = list(l = 130, b = 40))

# control text angle
heatmaply(mtcars, column_text_angle = 90, margins = c(40, 130))
# the same as using srtCol:
# heatmaply(mtcars, srtCol = 90) %>% layout(margin = list(l = 130, b = 40))



x <- mtcars
# different colors
```

```
heatmaply(x, colors = heat.colors(200))
# using special scale_fill_gradient_fun colors
heatmaply(x, scale_fill_gradient_fun = scale_color_gradient())


# We can join two heatmaps together:
library(heatmaply)
hm1 <- heatmaply(mtcars, margins = c(40, 130))
hm2 <- heatmaply(mtcars, scale = "col", margins = c(40, 130))
subplot(hm1, hm2, margin = .2)

# If we want to share the Y axis, then it is risky to keep any of the dendrograms:
library(heatmaply)
hm1 <- heatmaply(mtcars, Colv = FALSE, Rowv = FALSE, margins = c(40, 130))
hm2 <- heatmaply(mtcars,
  scale = "col", Colv = FALSE, Rowv = FALSE,
  margins = c(40, 130)
)
subplot(hm1, hm2, margin = .02, shareY = TRUE)


# We can save heatmaply as an HTML file by using:
heatmaply(iris[, -5], file = "heatmaply_iris.html")
# or a png/pdf/jpeg file using:
heatmaply(iris[, -5], file = "heatmaply_iris.png")
# or just doing it in one go:
heatmaply(iris[, -5], file = c("heatmaply_iris.html", "heatmaply_iris.png"))



# If we don't want the HTML to be selfcontained, we can use the following:
library(heatmaply)
library(htmlwidgets)
heatmaply(iris[, -5]) %>%
  saveWidget(file = "heatmaply_iris.html", selfcontained = FALSE)


# Example for using RowSideColors

x <- as.matrix(datasets::mtcars)
rc <- colorspace::rainbow_hcl(nrow(x))

library(gplots)
library(viridis)
heatmap.2(x,
  trace = "none", col = viridis(100),
  RowSideColors = rc
)

heatmaply(x,
  seriate = "mean",
  RowSideColors = rc
)
```

```
heatmaply(x[, -c(8, 9)],
  seriate = "mean",
  col_side_colors = c(rep(0, 5), rep(1, 4)),
  row_side_colors = x[, 8:9]
)
heatmaply(x[, -c(8, 9)],
  seriate = "mean",
  col_side_colors = data.frame(a = c(rep(0, 5), rep(1, 4))),
  row_side_colors = x[, 8:9]
)


## Example of using Rowv And Colv for custumized dendrograms.


x <- as.matrix(datasets::mtcars)

# now let's spice up the dendrograms a bit:
library(dendextend)

row_dend <- x %>%
  dist() %>%
  hclust() %>%
  as.dendrogram() %>%
  set("branches_k_color", k = 3) %>%
  set("branches_lwd", 4) %>%
  ladderize()
#    rotate_DendSer(ser_weight = dist(x))
col_dend <- x %>%
  t() %>%
  dist() %>%
  hclust() %>%
  as.dendrogram() %>%
  set("branches_k_color", k = 2) %>%
  set("branches_lwd", 4) %>%
  ladderize()
#    rotate_DendSer(ser_weight = dist(t(x)))

heatmaply(x, Rowv = row_dend, Colv = col_dend)


heatmaply(is.na10(airquality))
heatmaply(is.na10(airquality), grid_gap = 1)

# grid_gap can handle quite large data matrix
heatmaply(matrix(1:10000, 100, 100), k_row = 3, k_col = 3, grid_gap = 1)

# Examples of playing with font size:
heatmaply(mtcars, fontsize_col = 20, fontsize_row = 5, margin = c(100, 90))
```

```
# Example for using subplot_width/subplot_height

heatmaply(percentize(mtcars),
  subplot_widths = c(0.6, 0.4),
  subplot_heights = c(0.05, 0.95)
)



# Example of removing labels and thus making the plot faster
heatmaply(iris, showticklabels = c(TRUE, FALSE), margins = c(80, 10))

# this is what allows for a much larger matrix to be printed:
set.seed(2017 - 05 - 18)
large_x <- matrix(rnorm(19), 1000, 100)
heatmaply(large_x, dendrogram = FALSE, showticklabels = FALSE, margins = c(1, 1))

## End(Not run)
## Not run:
heatmaply_na(airquality)

## End(Not run)
## Not run:
heatmaply_cor(cor(mtcars))

## End(Not run)
```

`heatmapr` *Creates a heatmapr object*

Description

An object of class heatmapr includes all the needed information for producing a heatmap. The goal is to separate the pre-processing of the heatmap elements from the graphical rendering of the object, which could be done using plotly (but potentially also with other graphical devices).

Usage

```
heatmapr(
  x,
  Rowv = NULL,
  Colv = NULL,
  distfun = dist,
  hclustfun = hclust,
  dist_method = NULL,
  hclust_method = NULL,
  distfun_row = distfun,
  hclustfun_row = hclustfun,
```

```
  distfun_col = distfun,
  hclustfun_col = hclustfun,
  dendrogram = c("both", "row", "column", "none"),
  show_dendrogram = c(TRUE, TRUE),
  reorderfun = function(d, w) reorder(d, w),
  k_row = 1,
  k_col = 1,
  symm = FALSE,
 revC = symm || (is.dendrogram(Colv) & is.dendrogram(Rowv) & identical(Rowv, rev(Colv))),
  scale = c("none", "row", "column"),
  na.rm = TRUE,
  labRow = rownames(x),
  labCol = colnames(x),
  cexRow = NULL,
  cexCol = NULL,
  digits = 3L,
  cellnote = NULL,
  theme = NULL,
  colors = "RdYlBu",
  width = NULL,
  height = NULL,
  xaxis_height = 80,
  yaxis_width = 120,
  xaxis_font_size = NULL,
  yaxis_font_size = NULL,
  brush_color = "#0000FF",
  show_grid = TRUE,
  anim_duration = 500,
  row_side_colors = NULL,
  col_side_colors = NULL,
  seriate = c("OLO", "mean", "none", "GW"),
  point_size_mat = NULL,
  custom_hovertext = NULL,
  ...
)
```

Arguments

x	A numeric matrix Defaults to TRUE unless x contains any NAs.
Rowv	determines if and how the row dendrogram should be reordered. By default, it is TRUE, which implies dendrogram is computed and reordered based on row means. If NULL or FALSE, then no dendrogram is computed and no reordering is done. If a dendrogram (or hclust), then it is used "as-is", ie without any reordering. If a vector of integers, then dendrogram is computed and reordered based on the order of the vector.
Colv	determines if and how the column dendrogram should be reordered. Has the options as the Rowv argument above and additionally when x is a square matrix, Colv = "Rowv" means that columns should be treated identically to the rows.

distfun	function used to compute the distance (dissimilarity) between both rows and columns. Defaults to dist.
hclustfun	function used to compute the hierarchical clustering when Rowv or Colv are not dendrograms. Defaults to hclust.
dist_method	default is NULL (which results in "euclidean" to be used). Can accept alternative character strings indicating the method to be passed to distfun. By default distfun. is dist hence this can be one of "euclidean", "maximum", "manhattan", "canberra", "binary" or "minkowski".
hclust_method	default is NULL (which results in "complete" to be used). Can accept alternative character strings indicating the method to be passed to hclustfun By default hclustfun is hclust hence this can be one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).
distfun_row	distfun for row dendrogram only.
hclustfun_row	hclustfun for col dendrogram only.
distfun_col	distfun for row dendrogram only.
hclustfun_col	hclustfun for col dendrogram only.
dendrogram	character string indicating whether to compute 'none', 'row', 'column' or 'both' dendrograms. Defaults to 'both'. However, if Rowv (or Colv) is FALSE or NULL and dendrogram is 'both', then a warning is issued and Rowv (or Colv) arguments are honoured.
show_dendrogram	Logical vector of length controlling whether the row and column dendrograms are displayed. If a logical scalar is provided, it is repeated to become a logical vector of length two.
reorderfun	function(d, w) of dendrogram and weights for reordering the row and column dendrograms. The default uses reorder.dendrogram
k_row	an integer scalar with the desired number of groups by which to color the dendrogram's branches in the rows (uses color_branches) If NA then find_k is used to deduce the optimal number of clusters.
k_col	an integer scalar with the desired number of groups by which to color the dendrogram's branches in the columns (uses color_branches) If NA then find_k is used to deduce the optimal number of clusters.
symm	logical indicating if x should be treated symmetrically; can only be true when x is a square matrix.
revC	logical indicating if the column order should be reversed for plotting. Default (when NULL) - is FALSE, unless symm is TRUE. This is useful for cor matrix.
scale	character indicating if the values should be centered and scaled in either the row direction or the column direction, or none. The default is "none".
na.rm	logical indicating whether NA's should be removed.
labRow	character vectors with row labels to use (from top to bottom); default to rownames(x).
labCol	character vectors with column labels to use (from left to right); default to colnames(x).

cexRow	positive numbers. If not NULL, it will override xaxis_font_size and will give it a value cexRow*14
cexCol	positive numbers. If not NULL, it will override yaxis_font_size and will give it a value cexCol*14
digits	integer indicating the number of decimal places to be used by round for 'label'.
cellnote	(optional) matrix of the same dimensions as x that has the human-readable version of each value, for displaying on top of the heatmap cells.
theme	A custom CSS theme to use. Currently the only valid values are "" and "dark". "dark" is primarily intended for standalone visualizations, not R Markdown or Shiny.
colors	Either a colorbrewer2.org palette name (e.g. "YlOrRd" or "Blues"), or a vector of colors to interpolate in hexadecimal "#RRGGBB" format, or a color interpolation function like colorRamp.
width	Width in pixels (optional, defaults to automatic sizing).
height	Height in pixels (optional, defaults to automatic sizing).
xaxis_height	Size of axes, in pixels.
yaxis_width	Size of axes, in pixels.
xaxis_font_size	Font size of axis labels, as a CSS size (e.g. "14px" or "12pt").
yaxis_font_size	Font size of axis labels, as a CSS size (e.g. "14px" or "12pt").
brush_color	The base color to be used for the brush. The brush will be filled with a low-opacity version of this color. "#RRGGBB" format expected.
show_grid	TRUE to show gridlines, FALSE to hide them, or a numeric value to specify the gridline thickness in pixels (can be a non-integer).
anim_duration	Number of milliseconds to animate zooming in and out. For large x it may help performance to set this value to 0.
row_side_colors, col_side_colors	data.frame of factors to produce row/column side colors in the style of heatmap.2/heatmap.3. col_side_colors should be "wide", ie be the same dimensions as the column side colors it will produce.
seriate	character indicating the method of matrix sorting (default: "OLO"). Implemented options include: "OLO" (Optimal leaf ordering, optimizes the Hamiltonian path length that is restricted by the dendrogram structure - works in O(n^4)) "mean" (sorts the matrix based on the reorderfun using marginal means of the matrix. This is the default used by heatmap.2), "none" (the default order produced by the dendrogram), "GW" (Gruvaeus and Wainer heuristic to optimize the Hamiltonian path length that is restricted by the dendrogram structure)
point_size_mat	A matrix of values which can be mapped to point size
custom_hovertext	Custom hovertext matrix (the same dimensions as the input).
...	currently ignored

Source

The interface was designed based on heatmap, heatmap.2, and (the also d3heatmap).

See Also

heatmap, heatmap.2

Examples

```
## Not run:
library(heatmaply)
hm <- heatmapr(mtcars, scale = "column", colors = "Blues")
heatmaply(hm)

## End(Not run)
```

---

is.heatmapr *Is the object of class heatmapr*

Description

Is the object of class heatmapr.

Usage

```
is.heatmapr(x)
```

Arguments

x	an object.

Value

logical - is the object of class heatmapr.

`is.na10` *Indicates which elements are missing (either 1 and 0)*

Description

is.na10 is a helper function for creating heatmaps to diagnose missing value patterns. It is similar to is.na but instead of returning a logical TRUE/FALSE vector (or matrix) it returns a numeric 1/0 output. This enables the heatmaply function to be used on the data.

Usage

```
is.na10(x, ...)
```

Arguments

`x`	a vector, matrix or data.frame.
`...`	not used.

Value

Returns a numeric (instead of a logical) variable/matrix of 1 (missing) or 0 (not missing) values (hence the name is.na10) while still preserving the attributes resulted from running is.na.

These are useful for funnelling into a heatmap (see the examples).

See Also

is.na, the grid_gap parameter in heatmaply.

Examples

```
## Not run:
x <- mtcars
x <- data.frame(x)
x$am <- factor(x$am)
x$vs <- factor(x$vs)
set.seed(2017 - 01 - 19)
x[sample(nrow(x))[1:6], sample(ncol(x))[1:6]] <- NA


# nice grey colors from here: https://github.com/njtierney/visdat/blob/master/R/vis_miss_ly.R
x %>%
  is.na10() %>%
  heatmaply(colors = c("grey80", "grey20"), dendrogram = "none")
x %>%
  is.na10() %>%
  heatmaply(colors = c("grey80", "grey20"), k_col = 2, k_row = 2)

heatmaply(is.na10(airquality),
```

```
  grid_gap = 1,
  colors = c("grey80", "grey20"), k_col = 2, k_row = 2
)

## End(Not run)
```

`is.plotly` *Checks if an object is of class plotly or not.*

Description

Helpful for the plot_method in heatmaply.

Usage

```
is.plotly(x)
```

Arguments

x	an object to check

Value

TRUE if the object inherits "plotly" as a class.

`normalize` *Normalization transformation (0-1)*

Description

An Empirical Normalization Transformation brings data to the 0 to 1 scale by substracting the minimum and dividing by the maximum of all observations. This is similar to percentize in that it allows to compare variables of different scales, but it also keeps the shape of the distribution.

Usage

```
normalize(x, ...)
```

Arguments

x	a vector or a data.frame.
...	Currently ignored.

Value

A vector (or data.frame) after normalizing the numeric variables.

See Also

percentize

Examples

```
## Not run:
x <- mtcars
x <- data.frame(x)
x$am <- factor(x$am)
x$vs <- factor(x$vs)
heatmaply(percentize(x))
heatmaply(normalize(x))


x <- data.frame(a = 1:10, b = 11:20)
x[4:6, 1:2] <- NA
normalize(x)
normalize(x[, 1])

## End(Not run)
```

`percentize` *Empirical Percentile Transformation*

Description

An Empirical Percentile Transformation (percentize) is similar to taking the rank of a variable. The difference is that it is simpler to compare and interpret the transformed variables.

This is helpful for comparing several variables in a heatmap (e.g.: heatmaply).

Usage

```
percentize(x, ...)
```

Arguments

`x`	a vector or a data.frame.
`...`	Currently ignored.

Value

A vector (or data.frame) after ecdf was used on that vector. If x is a data.frame then only the numeric variables are transformed.

See Also

normalize

Examples

```
## Not run:
x <- mtcars
x <- data.frame(x)
x$am <- factor(x$am)
x$vs <- factor(x$vs)
heatmaply(percentize(x))


x <- data.frame(a = 1:10, b = 11:20)
x[4:6, 1:2] <- NA
percentize(x)
percentize(x[, 1])

## End(Not run)
```

`RColorBrewer_colors` *RColorBrewer color Ramp Palette*

Description

Functions for getting the colors of RColorBrewer (i.e.: brewer.pal) without the limitation of only 9/11 color values, based on colorRampPalette.

For sequential palettes this is not essential since we have viridis. But for diverging palettes this is quit essential.

The sequential palettes names are Blues BuGn BuPu GnBu Greens Greys Oranges OrRd PuBu PuBuGn PuRd Purples RdPu Reds YlGn YlGnBu YlOrBr YlOrRd

The diverging palettes are BrBG PiYG PRGn PuOr RdBu RdGy RdYlBu RdYlGn Spectral And also cool_warm. The cool_warm palette is based on Kenneth Moreland's proposal (see ref). It goes from blue (cool) to ref (warm), based on well thought-out design elements.

Usage

```
BrBG(n)

PiYG(n)

PRGn(n)

PuOr(n)

RdBu(n)

RdGy(n)

RdYlBu(n)
```

```
RdYlGn(n)

Spectral(n)

Blues(n)

BuGn(n)

BuPu(n)

GnBu(n)

Greens(n)

Greys(n)

Oranges(n)

OrRd(n)

PuBu(n)

PuBuGn(n)

PuRd(n)

Purples(n)

RdPu(n)

Reds(n)

YlGn(n)

YlGnBu(n)

YlOrBr(n)

YlOrRd(n)

cool_warm(n)
```

Arguments

n	the number of colors (>= 1) to be in the palette.

Value

A character vector of color names.

References

* Moreland, Kenneth. "Diverging color maps for scientific visualization." Advances in Visual Computing (2009): 92-103. url: http://www.kennethmoreland.com/color-maps/ The code was provided here: http://stackoverflow.com/a/44073011/256662 Thanks to the user YAK, who relied on the code from the Rgnuplot package (which is duplicated here, in order to save the need to import the entire package)

Examples

```
## Not run:

library(RColorBrewer)
display.brewer.all(n = 11, type = "div")
title(main = "Divergent color palette")
display.brewer.all(n = 9, type = c("seq"))
title(main = "Sequential color palette")



img <- function(obj, nam) {
  image(1:length(obj), 1, as.matrix(1:length(obj)),
    col = obj,
    main = nam, ylab = "", xaxt = "n", yaxt = "n", bty = "n"
  )
}

par(mfrow = c(10, 1))
img(rev(cool_warm(500)), "cool_warm, (Moreland 2009)")
img(RdBu(500), "RdBu")
img(BrBG(500), "BrBG")
img(PiYG(500), "PiYG")
img(PRGn(500), "PRGn")
img(PuOr(500), "PuOr")
img(RdGy(500), "RdGy")
img(RdYlBu(500), "RdYlBu")
img(RdYlGn(500), "RdYlGn")
img(Spectral(500), "Spectral")


library(heatmaply)
heatmaply(cor(mtcars), colors = PiYG, limits = c(-1, 1))
heatmaply(cor(mtcars), colors = RdBu, limits = c(-1, 1))

## End(Not run)
```

Index