Package: ggcorrheatmap (via r-universe)

June 5, 2026

Title Make Flexible 'ggplot2' Correlation Heatmaps

Version 0.3.0.9000

Description Create correlation heatmaps with 'ggplot2' and customise them with flexible annotation and clustering. Symmetric heatmaps can use triangular or mixed layouts, removing redundant information or displaying complementary information in the two halves. There is also support for general heatmaps not displaying correlations.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

URL https://github.com/leod123/ggcorrheatmap, https://leod123.github.io/ggcorrheatmap/

BugReports https://github.com/leod123/ggcorrheatmap/issues

Imports ggplot2, scales, dplyr, dendextend, ggnewscale, stats, rlang (>= 1.1.0), cli

Suggests testthat (>= 3.0.0), vdiffr

Config/testthat/edition 3

Config/Needs/website rmarkdown, patchwork, cowplot, tibble, tidyr

Repository https://leod123.r-universe.dev

Date/Publication 2025-10-07 08:27:02 UTC

RemoteUrl https://github.com/leod123/ggcorrheatmap

RemoteRef HEAD

RemoteSha 99f2b4296ea44c3abb5f57e444d49ec3ae157731

Contents

`add_mixed_layout` *Add a layout column to long format data for mixed layouts.*

Description

Add a layout column to long format data for mixed layouts.

Usage

```
add_mixed_layout(
  x,
  rows = "row",
  cols = "col",
  values = "value",
  layout,
  name = "layout"
)
```

Arguments

`x`	Long format data frame of a symmetric matrix.
`rows`, `cols`, `values`	Columns containing rows, columns, and values.
`layout`	Character vector of length two with a mixed layout (two opposing triangles).
`name`	Name of the column that should contain the layouts.

Value

The input data frame with a new column added, showing in which triangle each value would be in a mixed layout.

Examples

```
# Make long format symmetric data
long_df <- data.frame(rw = rep(letters[1:4], 4),
                      cl = rep(letters[1:4], each = 4),
                      val = 0)

long_df <- add_mixed_layout(long_df, rw, cl, val,
                            layout = c("topleft", "bottomright"))

head(long_df)
```

---

cor_long *Make a correlation matrix from long format data.*

Description

Make a correlation matrix from long format data.

Usage

```
cor_long(
  x,
  rows,
  cols,
  values,
  y = NULL,
  rows2 = NULL,
  cols2 = NULL,
  values2 = NULL,
  out_format = c("long", "wide"),
  method = "pearson",
  use = "everything",
  p_values = FALSE,
  p_adjust = "none",
  p_thresholds = c(`***` = 0.001, `**` = 0.01, `*` = 0.05, 1),
  p_sym_add = NULL,
  p_sym_digits = 2
)
```

Arguments

x	A long format data frame containing the data to correlate.
rows, cols	The columns in x containing the values that should be in the rows and columns of the correlation matrix.
values	Name of the column in x containing the values of the correlation matrix.
y	Optional second data frame for correlating with the data frame from x.

rows2, cols2	Optional names of columns with values for the rows and columns of a second matrix (taken from y).
values2	Optional column for the values of a second matrix.
out_format	Format of output correlation matrix ("long" or "wide").
method	Correlation method given to stats::cor().
use	Missing value strategy of stats::cor().
p_values	Logical indicating if p-values should be calculated.
p_adjust	String specifying the multiple testing adjustment method to use for the p-values (default is "none"). Passed to stats::p.adjust().
p_thresholds	Named numeric vector specifying p-value thresholds (in ascending order) to mark. The last element must be 1 or higher (to set the upper limit). Names must be unique, but one element can be left unnamed (by default 1 is unnamed, meaning values between the threshold closest to 1 and 1 are not marked in the plot). If NULL, no thresholding is done and p-value intervals are not marked with symbols.
p_sym_add	String with the name of the column to add to p-value symbols from p_thresholds (one of 'values', 'p_val', 'p_adj'). NULL (default) results in just the symbols.
p_sym_digits	Number of digits to use for the column in p_sym_add.

Details

If there is only one input data frame (x), a wide matrix is constructed from x and passed to stats::cor(), resulting in a correlation matrix with the column-column correlations.

If y is a data frame and rows2, cols2 and values2 are specified, the wide versions of x and y are correlated (stats::cor(wide_x, wide_y)) resulting in a correlation matrix with the columns of x in the rows and the columns of y in the columns.

Value

A correlation matrix (if wide format) or a long format data frame with the columns 'row', 'col', and 'value' (containing correlations).

Examples

```
set.seed(123)
cor_in <- data.frame(row = rep(letters[1:10], each = 5),
                     col = rep(LETTERS[1:5], 10),
                     val = rnorm(50))
# Wide format output (default)
corr_wide <- cor_long(cor_in, row, col, val)

# Long format output
corr_long <- cor_long(cor_in, row, col, val,
                      out_format = "long")

# Correlation between two matrices
cor_in2 <- data.frame(rows = rep(letters[1:10], each = 10),
```

```
                      cols = rep(letters[1:10], 10),
                      values = rnorm(100))
corr2 <- cor_long(cor_in, row, col, val,
                  cor_in2, rows, cols, values)
```

---

ggcorrhm *Make a correlation heatmap with ggplot2.*

Description

Make a correlation heatmap from input matrices. Uses a diverging colour scale centered around 0.

Usage

```
ggcorrhm(
  x,
  y = NULL,
  cor_method = "pearson",
  cor_use = "everything",
  cor_in = FALSE,
  high = "sienna2",
  mid = "white",
  low = "skyblue2",
  midpoint = 0,
  limits = c(-1, 1),
  bins = NULL,
  layout = "full",
  mode = if (length(layout) == 1) "heatmap" else c("heatmap", "text"),
  include_diag = TRUE,
  split_diag = FALSE,
  na_col = "grey50",
  na_remove = FALSE,
  return_data = FALSE,
  col_scale = NULL,
  col_name = NULL,
  size_range = c(4, 10),
  size_scale = NULL,
  size_name = NULL,
  legend_order = NULL,
  p_values = FALSE,
  p_adjust = "none",
  p_thresholds = c(`***` = 0.001, `**` = 0.01, `*` = 0.05, 1),
  cell_labels = FALSE,
  cell_label_p = FALSE,
  cell_label_col = "black",
  cell_label_size = 3,
```

```
  cell_label_digits = 2,
  cell_bg_col = "white",
  cell_bg_alpha = 0,
  border_col = "grey",
  border_lwd = 0.1,
  border_lty = 1,
  show_names_diag = TRUE,
  names_diag_params = NULL,
  show_names_rows = FALSE,
  names_rows_side = "left",
  show_names_cols = FALSE,
  names_cols_side = "top",
  show_names_x = NULL,
  names_x_side = NULL,
  show_names_y = NULL,
  names_y_side = NULL,
  annot_rows_df = NULL,
  annot_cols_df = NULL,
  annot_rows_col = NULL,
  annot_cols_col = NULL,
  annot_rows_side = "right",
  annot_cols_side = "bottom",
  annot_dist = 0.2,
  annot_gap = 0,
  annot_size = 0.5,
  annot_border_col = if (length(border_col) == 1) border_col else "grey",
  annot_border_lwd = if (length(border_lwd) == 1) border_lwd else 0.5,
  annot_border_lty = if (length(border_lty) == 1) border_lty else 1,
  annot_na_col = na_col,
  annot_na_remove = na_remove,
  annot_rows_params = NULL,
  annot_cols_params = NULL,
  show_annot_names = TRUE,
  annot_names_size = 3,
  annot_rows_names_side = "bottom",
  annot_cols_names_side = "left",
  annot_rows_names_params = NULL,
  annot_cols_names_params = NULL,
  annot_rows_name_params = NULL,
  annot_cols_name_params = NULL,
  cluster_rows = FALSE,
  cluster_cols = FALSE,
  cluster_distance = "euclidean",
  cluster_method = "complete",
  show_dend_rows = TRUE,
  show_dend_cols = TRUE,
  dend_rows_side = "right",
  dend_cols_side = "bottom",
```

```
  dend_col = "black",
  dend_dist = 0,
  dend_height = 0.3,
  dend_lwd = 0.3,
  dend_lty = 1,
  dend_rows_params = NULL,
  dend_cols_params = NULL,
  dend_rows_extend = NULL,
  dend_cols_extend = NULL,
  split_rows = NULL,
  split_cols = NULL,
  split_rows_side = "right",
  split_cols_side = "bottom"
)
```

Arguments

x	Matrix or data frame in wide format containing the columns to correlate against each other or against the columns in y.
y	Optional matrix or data frame in wide format containing columns to correlate with the columns in x.
cor_method	String specifying correlation method to use in the stats::cor() function. Default is 'pearson'.
cor_use	String specifying the use argument of stats::cor(), which defines how to deal with missing values. Default is 'everything'.
cor_in	Logical indicating if the input data contains correlation values and any correlation computations (including p-values) should be skipped. Default is FALSE.
high	Colour to use for the highest value of the colour scale.
mid	Colour to use for 0 in the colour scale.
low	Colour to use for the lowest value of the colour scale.
midpoint	Value for the middle point of the colour scale.
limits	Numeric vector of length two for the limits of the colour scale. NULL uses the default.
bins	Number of bins to divide the scale into (if continuous values). A 'double' class value uses 'nice.breaks' to put the breaks at nice numbers which may not result in the specified number of bins. If an integer the number of bins will be prioritised.
layout	String specifying the layout of the output heatmap. Possible layouts include 'topleft', 'topright', 'bottomleft', 'bottomright', or the 'whole'/'full' heatmap (default and only possible option if the matrix is not square with identical dimnames). A combination of the first letters of each word also works (i.e. f, w, tl, tr, bl, br). If layout is of length two with two opposing triangles, a mixed layout will be used. For mixed layouts, mode needs a vector of length two (applied in the same order as layout). See details of gghm() for more information.

mode	A string specifying plotting mode. Possible values are heatmap/hm for a normal heatmap, a number from 1 to 25 to draw the corresponding shape, text to write the cell values instead of filling cells (colour scaling with value), and none for blank cells.
include_diag	Logical indicating if the diagonal cells should be plotted (if the matrix is square with the same row- and colnames).
split_diag	Logical indicating if the diagonal cells should be drawn as triangles, splitting the diagonal in two.
na_col	Colour to use for cells with NA (both main heatmap and annotation).
na_remove	Logical indicating if NA values in the heatmap should be omitted (meaning no cell border is drawn). This does not affect how NAs are handled in the correlation computations, use the cor_use argument for NA handling in correlation.
return_data	Logical indicating if the data used for plotting (i.e. the correlation values and, if computed, clustering and p-values) should be returned.
col_scale	Scale to use for cell colours. If NULL (default), a divergent scale is constructed from the high, mid, low, midpoint, limits, and bins arguments. These arguments are ignored if a ggplot2::scale_* function is provided instead. If a string, the corresponding Brewer or Viridis scale is used. A string with a scale name with "rev_" in the beginning or "_rev" at the end will result in the reversed scale. In mixed layouts, can also be a list of length two containing the two scales to use.
col_name	String to use for the correlation scale. If NULL (default) the text will depend on the correlation method. Can be two values in mixed layouts for dual scales.
size_range	Numeric vector of length 2, specifying lower and upper ranges of shape sizes. Ignored if size_scale is not NULL.
size_scale	ggplot2::scale_size_* call to use for size scaling if mode is a number from 1 to 25 (R pch). The default behaviour (NULL) is to use a continuous scale with the absolute values of the correlation.
size_name	String to use for the size scale legend title. Can be two values in mixed layouts for dual scales.
legend_order	Integer vector specifying the order of legends (first value is for the first legend, second for the second, etc). The default (NULL) shows all but size legends. NAs hide the corresponding legends, a single NA hides all. Ignored for ggplot2 scale objects in col_scale and size_scale.
p_values	Logical indicating if p-values should be calculated. Use with p_thresholds to mark cells, and/or return_data to get the p-values in the output data.
p_adjust	String specifying the multiple testing adjustment method to use for the p-values (default is "none"). Passed to stats::p.adjust().
p_thresholds	Named numeric vector specifying p-value thresholds (in ascending order) to mark. The last element must be 1 or higher (to set the upper limit). Names must be unique, but one element can be left unnamed (by default 1 is unnamed, meaning values between the threshold closest to 1 and 1 are not marked in the plot). If NULL, no thresholding is done and p-value intervals are not marked with symbols.

`cell_labels` Logical specifying if the cells should be labelled with the correlation values. Alternatively, a matrix or data frame with the same shape and dimnames as `x` containing values to write in the cells. If mode is `text`, the cell label colours will scale with the correlation values and `cell_label_col` is ignored.

`cell_label_p` Logical indicating if, when `cell_labels` is `TRUE`, p-values should be written instead of correlation values.

`cell_label_col` Colour to use for cell labels, passed to `ggplot2::geom_text()`.

`cell_label_size`
Size of cell labels, used as the `size` argument in `ggplot2::geom_text()`.

`cell_label_digits`
Number of digits to display when cells are labelled (if numeric values). Default is 2, passed to `base::round()`. NULL for no rounding.

`cell_bg_col` Colour to use for cell backgrounds in modes 'text' and 'none'.

`cell_bg_alpha` Alpha for cell colours in modes 'text' and 'none'.

`border_col` Colour of cell borders. If `mode` is not a number, `border_col` can be set to NA to remove borders completely.

`border_lwd` Size of cell borders. If `mode` is a number, `border_col` can be set to 0 to remove borders.

`border_lty` Line type of cell borders. Either a number or its corresponding name, or a string of length 2, 4, 6, or 8. See 'lty' of `graphics::par()` for details. Not supported for numeric `mode`.

`show_names_diag`
Logical indicating if names should be written in the diagonal cells.

`names_diag_params`
List with named parameters (such as size, angle, etc) passed on to geom_text when writing the column names in the diagonal.

`show_names_rows`, `show_names_cols`
Logical indicating if row/colnames should be written on the axes. Labels can be customised using `ggplot2::theme()` on the output plot.

`names_rows_side`
String specifying position of the row names ("left" or "right").

`names_cols_side`
String specifying position of the column names ("top" or "bottom").

`show_names_x`, `show_names_y`
Deprecated in favour of `show_names_rows` and `show_names_cols`. Logical indicating if row/colnames should be shown.

`names_x_side` Deprecated in favour of `names_cols_side`. String specifying position of the x axis names ("top" or "bottom").

`names_y_side` Deprecated in favour of `names_rows_side`. String specifying position of the y axis names ("left" or "right").

`annot_rows_df`, `annot_cols_df`
Data frame for row and column annotations. The names of the columns in the data must be included, either as row names or in a column named `.names`. Each other column specifies an annotation where the column name will be used as the

annotation name (in the legend and next to the annotation). Numeric columns will use a continuous colour scale while factor or character columns use discrete scales.

`annot_rows_col`, `annot_cols_col`
: Named list for row and column annotation colour scales. The names should specify which annotation each scale applies to. Elements can be strings or ggplot2 "Scale" class objects. If a string, it is used as the brewer palette or viridis option. If a scale object it is used as is, allowing more flexibility. This may change the order that legends are drawn in, specify order using the `guide` argument in the `ggplot2` scale function.

`annot_rows_side`
: String specifying which side row annotation should be drawn ('left' or 'right', defaults to 'left').

`annot_cols_side`
: String specifying which side column annotation should be drawn ('bottom' or 'top', defaults to 'bottom').

`annot_dist`
: Distance between heatmap and first annotation cell where 1 is the size of one heatmap cell. Used for both row and column annotation.

`annot_gap`
: Distance between each annotation where 1 is the size of one heatmap cell. Used for both row and column annotation.

`annot_size`
: Size (width for row annotation, height for column annotation) of annotation cells compared to a heatmap cell. Used for both row and column annotation.

`annot_border_col`
: Colour of cell borders in annotation. By default it is the same as `border_col` of the main heatmap if it is of length 1, otherwise uses default (grey).

`annot_border_lwd`
: Line width of cell borders in annotation. By default it is the same as `border_lwd` of the main heatmap if it is of length 1, otherwise uses default (0.5).

`annot_border_lty`
: Line type of cell borders in annotation. By default it is the same as `border_lty` of the main heatmap if it is of length 1, otherwise uses default (solid).

`annot_na_col`
: Colour to use for NA values in annotations. Annotation-specific colour can be set in the ggplot2 scales in the `annot_*_fill` arguments.

`annot_na_remove`
: Logical indicating if NAs in the annotations should be removed (producing empty spaces).

`annot_rows_params`, `annot_cols_params`
: Named list with parameters for row or column annotations to overwrite the defaults set by the `annot_*` arguments, each name corresponding to the `*` part (see details of `gghm()` for more information).

`show_annot_names`
: Logical controlling if names of annotations should be shown in the drawing area.

`annot_names_size`
: Size of annotation names.

`annot_rows_names_side`
: String specifying which side the row annotation names should be on. Either "top" or "bottom".

annot_cols_names_side
: String specifying which side the column annotation names should be on. Either "left" or "right".

annot_rows_names_params, annot_cols_names_params
: Named list of parameters for row and column annotation names. Given to `ggplot2::geom_text()`.

annot_rows_name_params, annot_cols_name_params
: Deprecated and kept for backward compatibility. Named list of parameters given to `grid::textGrob()` for annotation names. Does not work well with heatmap splits.

cluster_rows, cluster_cols
: Logical indicating if rows or columns should be clustered. Can also be `hclust` or `dendrogram` objects.

cluster_distance
: String with the distance metric to use for clustering, given to `stats::dist()`.

cluster_method
: String with the clustering method to use, given to `stats::hclust()`.

show_dend_rows, show_dend_cols
: Logical indicating if a dendrogram should be drawn for the rows or columns.

dend_rows_side
: Which side to draw the row dendrogram on ('left' or 'right', defaults to 'left').

dend_cols_side
: Which side to draw the column dendrogram on ('bottom' or 'top', defaults to 'bottom').

dend_col
: Colour to use for dendrogram lines, applied to both row and column dendrograms.

dend_dist
: Distance from heatmap (or annotation) to leaves of dendrogram, measured in heatmap cells (1 is the size of one cell).

dend_height
: Number by which to scale dendrogram height, applied to both row and column dendrograms.

dend_lwd
: Linewidth of dendrogram lines, applied to both row and column dendrograms.

dend_lty
: Dendrogram line type, applied to both row and column dendrograms.

dend_rows_params, dend_cols_params
: Named list for row or column dendrogram parameters. See details of `gghm()` for more information.

dend_rows_extend, dend_cols_extend
: Named list or functional sequence for specifying `dendextend` functions to apply to the row or column dendrogram. See details of `gghm()` and `ggcorrhm()` for usage.

split_rows, split_cols
: Vectors for splitting the rows and columns into facets. Can be a numeric vector shorter than the number of rows/columns to split the heatmap after those indices, or a vector of the same length as the number of rows/columns containing the facet memberships. In the latter case names can be used to match with rows/columns. Alternatively, if clustering is applied a single numeric value is accepted for the number of clusters to divide the plot into.

split_rows_side, split_cols_side
: Which side the row/column facet strips should be drawn on ('left'/'right', 'top'/'bottom').

Details

ggcorrhm() makes it convenient to make correlation heatmaps, taking the input matrix or data frame to visualise the correlations between columns with the gghm() function. The input values can either be one matrix or data frame with columns to correlate with each other, or two matrices or data frames with columns to correlate between the matrices. No rownames are needed, but if two matrices are provided they should have the same number of rows and the rows should be ordered in a meaningful way (i.e. same sample/individual/etc in the same row in both).

Row and column names are displayed in the diagonal by default if the correlation matrix is square with identical dimnames (only x is provided or x and y are identical).

The colour scale is set to be a diverging gradient around 0, with options to change the low, mid, and high colours, the midpoint, and the limits (using the arguments of the same names). The bins argument converts the scale to a discrete scale divided into bins equally distributed bins (if an integer the breaks may be at strange numbers, if a double the number of bins may be different but the breaks are at nicer numbers). These arguments can be of length two (limits a list of length two) two apply to each triangle in a mixed layout (detailed more in the details section of gghm()). The size_range argument (for size scales) can also be a list of length two like limits.

The size scale, used when a numeric cell shape is specified, is set to vary the shape size between 4 and 10 (can be changed with the size_range argument) and to transform the values to absolute values (so that both positive and negative correlations are treated equally). This behaviour can be overwritten by setting size_scale to another ggplot2::scale_size_* function with the desired arguments, or ggplot2::scale_size() for no special behaviour. ggplot2::scale_size_area() also scales with the absolute value, but only the upper size limit can be set. When the absolute value transformation is used the legend for sizes loses its meaning (only displaying positive values) and is therefore set to not be shown if legend_order is NULL.

For square correlation matrices, the dendrogram customisation arguments dend_rows_extend and dend_cols_extend work best with functions that only change the dendrogram cosmetically such as the colours, linetypes or node shapes. While it is possible to reorder (using e.g. 'rotate', 'ladderize') or prune (using e.g. 'prune'), anything that changes the structure of the dendrogram may end up looking strange for square matrices if only applied to one dimension (e.g. the diagonal may not be on the diagonal, triangular or mixed layouts may not work). The same applies if the cluster_rows and cluster_cols arguments are hclust or dendrogram objects.

Value

The correlation heatmap as a ggplot object. If return_data is TRUE the output is a list containing the plot (named 'plot'), the correlations ('plot_data', with factor columns 'row' and 'col' and a column 'value' containing the cell values, and 'layout' which part of the heatmap each cell belongs to), and the result of the clustering ('row_clustering' and 'col_clustering', if clustered). If p-values were calculated, two additional columns named 'p_val' and 'p_adj' are included in 'plot_data', containing nominal and adjusted p-values.

Examples

```
# Basic usage
ggcorrhm(mtcars)

# With two matrices
```

```
ggcorrhm(iris[1:32, -5], mtcars)

# Different layout
ggcorrhm(mtcars, layout = "br")

# With clustering
ggcorrhm(mtcars, layout = "tl", cluster_rows = TRUE, cluster_cols = TRUE)

# With annotation
set.seed(123)
annot <- data.frame(.names = colnames(mtcars),
                    annot1 = rnorm(ncol(mtcars)),
                    annot2 = sample(letters[1:3], ncol(mtcars), TRUE))
ggcorrhm(mtcars, layout = "tr", annot_cols_df = annot)

# Both
ggcorrhm(mtcars, layout = "full", cluster_rows = TRUE, cluster_cols = TRUE,
         annot_rows_df = annot[, -3], annot_cols_df = annot[, -2])

# Mixed layout
ggcorrhm(mtcars, layout = c("tl", "br"))

# Using different plotting modes
ggcorrhm(mtcars, layout = c("bl", "tr"), mode = c("21", "none"),
         cell_labels = c(FALSE, TRUE))

# Different colour scales and split diagonal
ggcorrhm(mtcars, layout = c("bl", "tr"), mode = c("hm", "hm"),
         col_scale = c("A", "G"), split_diag = TRUE,
         show_names_diag = FALSE, border_col = 1, border_lwd = 0.3)
```

`ggcorrhm_tidy` `ggcorrhm()` *for long format data.*

Description

`ggcorrhm()` for long format data.

Usage

```
ggcorrhm_tidy(
  x,
  rows,
  cols,
  values,
  annot_rows = NULL,
  annot_cols = NULL,
  labels = NULL,
```

```
  facet_rows = NULL,
  facet_cols = NULL,
  cor_in = TRUE,
  ...
)
```

Arguments

x	Data containing data to plot or to correlate.
rows, cols, values	Columns to use as rows, columns, and values in the plotted matrix (if cor_in is TRUE) or the matrix to compute correlations from (cor_in is FALSE).
annot_rows, annot_cols	Columns containing values for row and column annotations.
labels	Column to use for cell labels, NULL for no labels, or TRUE to use the cell values. If cor_in is FALSE, only NULL, TRUE or FALSE is supported.
facet_rows, facet_cols	Columns to use for row/column facets.
cor_in	Logical indicating if the values are correlation values (TRUE, default) or values to be correlated. See details for more information.
...	Additional arguments for ggcorrhm().

Details

If cor_in is TRUE (the default), ggcorrhm_tidy() behaves similarly to gghm_tidy() but with the colour scales and arguments of ggcorrhm() instead of gghm().

If cor_in FALSE, the data is converted to wide format and the column-column correlations are computed. This means that if non-square correlation matrices are to be plotted the correlations have to be computed in advance and plotted with cor_in as TRUE. Additionally, annot_rows and annot_cols will both use the cols column for names, and labels can only take TRUE or FALSE.

On the other hand, if cor_in is TRUE any computation of correlations is skipped, meaning that p-values cannot be computed and would have to be generated in advance and passed as cell labels.

Value

A ggplot2 object with the heatmap. If return_data is TRUE, plotting data is returned as well.

Examples

```
library(dplyr)
# Basic example with long format correlation data
# Make some correlation data in long format
cor_dat <- cor(mtcars)
hm_in <- data.frame(row = rep(colnames(cor_dat), ncol(cor_dat)),
                    col = rep(colnames(cor_dat), each = ncol(cor_dat)),
                    val = as.vector(cor_dat))

ggcorrhm_tidy(hm_in, row, col, val,
```

```
              # Indicate that the data consists of correlation coefficients
              cor_in = TRUE)

# Or let the function compute the correlations
# (this limits some other functionality, see details)
raw_dat <- data.frame(row = rep(rownames(mtcars), ncol(mtcars)),
                      col = rep(colnames(mtcars), each = nrow(mtcars)),
                      val = unlist(mtcars))
ggcorrhm_tidy(raw_dat, row, col, val, cor_in = FALSE)
```

---

gghm *Make a heatmap with ggplot2.*

Description

Make a heatmap of a matrix/data frame using ggplot2. Square matrices where the row- and column names are the same can use triangular layouts that either only show one triangle or plot different things in the different triangles.

Usage

```
gghm(
  x,
  layout = "full",
  mode = if (length(layout) == 1) "heatmap" else c("heatmap", "text"),
  scale_data = NULL,
  col_scale = NULL,
  col_name = "value",
  limits = NULL,
  bins = NULL,
  size_scale = NULL,
  size_name = "value",
  legend_order = NULL,
  include_diag = TRUE,
  split_diag = FALSE,
  show_names_diag = FALSE,
  names_diag_params = NULL,
  show_names_rows = TRUE,
  names_rows_side = "left",
  show_names_cols = TRUE,
  names_cols_side = "top",
  show_names_x = NULL,
  names_x_side = NULL,
  show_names_y = NULL,
  names_y_side = NULL,
  na_col = "grey50",
  na_remove = FALSE,
```

```
return_data = FALSE,
cell_labels = FALSE,
cell_label_col = "black",
cell_label_size = 3,
cell_label_digits = 2,
border_col = "grey",
border_lwd = 0.1,
border_lty = 1,
cell_bg_col = "white",
cell_bg_alpha = 0,
annot_rows_df = NULL,
annot_cols_df = NULL,
annot_rows_col = NULL,
annot_cols_col = NULL,
annot_rows_side = "right",
annot_cols_side = "bottom",
annot_dist = 0.2,
annot_gap = 0,
annot_size = 0.5,
annot_border_col = if (length(border_col) == 1) border_col else "grey",
annot_border_lwd = if (length(border_lwd) == 1) border_lwd else 0.5,
annot_border_lty = if (length(border_lty) == 1) border_lty else 1,
annot_na_col = na_col,
annot_na_remove = na_remove,
annot_rows_params = NULL,
annot_cols_params = NULL,
show_annot_names = TRUE,
annot_names_size = 3,
annot_rows_names_side = "bottom",
annot_cols_names_side = "left",
annot_rows_names_params = NULL,
annot_cols_names_params = NULL,
annot_rows_name_params = NULL,
annot_cols_name_params = NULL,
cluster_rows = FALSE,
cluster_cols = FALSE,
cluster_distance = "euclidean",
cluster_method = "complete",
show_dend_rows = TRUE,
show_dend_cols = TRUE,
dend_rows_side = "right",
dend_cols_side = "bottom",
dend_col = "black",
dend_dist = 0,
dend_height = 0.3,
dend_lwd = 0.3,
dend_lty = 1,
dend_rows_params = NULL,
```

```
  dend_cols_params = NULL,
  dend_rows_extend = NULL,
  dend_cols_extend = NULL,
  split_rows = NULL,
  split_cols = NULL,
  split_rows_side = "right",
  split_cols_side = "bottom"
)
```

Arguments

x	Matrix or data frame in wide format to make a heatmap of. If rownames are present they are used for the y axis labels, otherwise the row number is used. If a column named `.names` (containing unique row identifiers) is present it will be used as rownames.
layout	String specifying the layout of the output heatmap. Possible layouts include 'topleft', 'topright', 'bottomleft', 'bottomright', or the 'whole'/'full' heatmap (default and only possible option if the matrix is not square). A combination of the first letters of each word also works (i.e. f, w, tl, tr, bl, br). If layout is of length two with two opposing triangles, a mixed layout will be used. For mixed layouts, `mode` needs a vector of length two (applied in the same order as layout). See details for more information.
mode	A string specifying plotting mode. Possible values are `heatmap`/`hm` for a normal heatmap, a number from 1 to 25 to draw the corresponding shape, `text` to write the cell values instead of filling cells (colour scaling with value), and `none` for blank cells.
scale_data	Character string specifying scaling of the matrix. NULL or "none" for no scaling, "rows" for rows, and "columns" for columns. Can also be a substring of the beginning of the words.
col_scale	Colour scale to use for cells. If NULL, the default ggplot2 scale is used. If a string, the corresponding Brewer or Viridis scale is used. A string with a scale name with "rev_" in the beginning or "_rev" at the end will result in the reversed scale. Can also be a ggplot2 scale object to overwrite the scale. In mixed layouts, a list of two scales can be provided.
col_name	String to use for the colour scale legend title. Can be two values in mixed layouts for dual scales.
limits	Numeric vector of length two for the limits of the colour scale. NULL uses the default.
bins	Number of bins to divide the scale into (if continuous values). A 'double' class value uses 'nice.breaks' to put the breaks at nice numbers which may not result in the specified number of bins. If an integer the number of bins will be prioritised.
size_scale	`ggplot2::scale_size_*` call to use for size scaling if `mode` is a number from 1 to 25 (R pch). In mixed layouts, can also be a list of length two containing the two scales to use.
size_name	String to use for the size scale legend title. Can be two values in mixed layouts for dual scales.

legend_order
: Integer vector specifying the order of legends (first value is for the first legend, second for the second, etc). The default (NULL) shows all legends. NAs hide the corresponding legends, a single NA hides all. Ignored for ggplot2 scale objects in col_scale and size_scale.

include_diag
: Logical indicating if the diagonal cells (of a square matrix with identical dimnames) should be plotted. Mostly only useful for getting a cleaner look with symmetric correlation matrices with triangular layouts, where the diagonal is known to be 1.

split_diag
: Logical indicating if the diagonal cells should be drawn as triangles, splitting the diagonal in two.

show_names_diag
: Logical indicating if names should be written in the diagonal cells.

names_diag_params
: List with named parameters (such as size, angle, etc) passed on to geom_text when writing the column names in the diagonal.

show_names_rows, show_names_cols
: Logical indicating if row/colnames should be written on the axes. Labels can be customised using ggplot2::theme() on the output plot.

names_rows_side
: String specifying position of the row names ("left" or "right").

names_cols_side
: String specifying position of the column names ("top" or "bottom").

show_names_x, show_names_y
: Deprecated in favour of show_names_rows and show_names_cols. Logical indicating if row/colnames should be shown.

names_x_side
: Deprecated in favour of names_cols_side. String specifying position of the x axis names ("top" or "bottom").

names_y_side
: Deprecated in favour of names_rows_side. String specifying position of the y axis names ("left" or "right").

na_col
: Colour to use for cells with NA (both main heatmap and annotation).

na_remove
: Logical indicating if NA values in the heatmap should be omitted (meaning no cell border is drawn). If NAs are kept, the fill colour can be set in the ggplot2 scale.

return_data
: Logical indicating if the data used for plotting and clustering results should be returned.

cell_labels
: Logical specifying if the cells should be labelled with the values. Alternatively, a matrix or data frame with the same shape and dimnames as x containing values to write in the cells. If mode is text, the cell label colours will scale with the cell values and cell_label_col is ignored.

cell_label_col
: Colour to use for cell labels, passed to ggplot2::geom_text().

cell_label_size
: Size of cell labels, used as the size argument in ggplot2::geom_text().

cell_label_digits
: Number of digits to display when cells are labelled (if numeric values). Default is 2, passed to base::round(). NULL for no rounding.

`border_col` Colour of cell borders. If `mode` is not a number, `border_col` can be set to NA to remove borders completely.

`border_lwd` Size of cell borders. If `mode` is a number, `border_col` can be set to 0 to remove borders.

`border_lty` Line type of cell borders. Either a number or its corresponding name, or a string of length 2, 4, 6, or 8. See 'lty' of `graphics::par()` for details. Not supported for numeric `mode`.

`cell_bg_col` Colour to use for cell backgrounds in modes 'text' and 'none'.

`cell_bg_alpha` Alpha for cell colours in modes 'text' and 'none'.

`annot_rows_df, annot_cols_df`
Data frame for row and column annotations. The names of the columns in the data must be included, either as row names or in a column named `.names`. Each other column specifies an annotation where the column name will be used as the annotation name (in the legend and next to the annotation). Numeric columns will use a continuous colour scale while factor or character columns use discrete scales.

`annot_rows_col, annot_cols_col`
Named list for row and column annotation colour scales. The names should specify which annotation each scale applies to. Elements can be strings or ggplot2 "Scale" class objects. If a string, it is used as the brewer palette or viridis option. If a scale object it is used as is, allowing more flexibility. This may change the order that legends are drawn in, specify order using the `guide` argument in the `ggplot2` scale function.

`annot_rows_side`
String specifying which side row annotation should be drawn ('left' or 'right', defaults to 'left').

`annot_cols_side`
String specifying which side column annotation should be drawn ('bottom' or 'top', defaults to 'bottom').

`annot_dist` Distance between heatmap and first annotation cell where 1 is the size of one heatmap cell. Used for both row and column annotation.

`annot_gap` Distance between each annotation where 1 is the size of one heatmap cell. Used for both row and column annotation.

`annot_size` Size (width for row annotation, height for column annotation) of annotation cells compared to a heatmap cell. Used for both row and column annotation.

`annot_border_col`
Colour of cell borders in annotation. By default it is the same as `border_col` of the main heatmap if it is of length 1, otherwise uses default (grey).

`annot_border_lwd`
Line width of cell borders in annotation. By default it is the same as `border_lwd` of the main heatmap if it is of length 1, otherwise uses default (0.5).

`annot_border_lty`
Line type of cell borders in annotation. By default it is the same as `border_lty` of the main heatmap if it is of length 1, otherwise uses default (solid).

`annot_na_col` Colour to use for NA values in annotations. Annotation-specific colour can be set in the ggplot2 scales in the `annot_*_fill` arguments.

annot_na_remove
: Logical indicating if NAs in the annotations should be removed (producing empty spaces).

annot_rows_params, annot_cols_params
: Named list with parameters for row and column annotations to overwrite the defaults set by the annot_* arguments, each name corresponding to the * part (see details for more information).

show_annot_names
: Logical controlling if names of annotations should be shown in the drawing area.

annot_names_size
: Size of annotation names.

annot_rows_names_side
: String specifying which side the row annotation names should be on. Either "top" or "bottom".

annot_cols_names_side
: String specifying which side the column annotation names should be on. Either "left" or "right".

annot_rows_names_params, annot_cols_names_params
: Named list of parameters for row and column annotation names. Given to ggplot2::geom_text().

annot_rows_name_params, annot_cols_name_params
: Deprecated and kept for backward compatibility. Named list of parameters given to grid::textGrob() for annotation names. Does not work well with heatmap splits.

cluster_rows, cluster_cols
: Logical indicating if rows or columns should be clustered. Can also be hclust or dendrogram objects.

cluster_distance
: String with the distance metric to use for clustering, given to stats::dist().

cluster_method
: String with the clustering method to use, given to stats::hclust().

show_dend_rows, show_dend_cols
: Logical indicating if a dendrogram should be drawn for the rows or columns.

dend_rows_side
: Which side to draw the row dendrogram on ('left' or 'right', defaults to 'left').

dend_cols_side
: Which side to draw the column dendrogram on ('bottom' or 'top', defaults to 'bottom').

dend_col
: Colour to use for dendrogram lines, applied to both row and column dendrograms.

dend_dist
: Distance from heatmap (or annotation) to leaves of dendrogram, measured in heatmap cells (1 is the size of one cell).

dend_height
: Number by which to scale dendrogram height, applied to both row and column dendrograms.

dend_lwd
: Linewidth of dendrogram lines, applied to both row and column dendrograms.

dend_lty
: Dendrogram line type, applied to both row and column dendrograms.

dend_rows_params, dend_cols_params
: Named list for row or column dendrogram parameters to overwrite common parameter values. See details for more information.

dend_rows_extend, dend_cols_extend
: Named list or functional sequence for specifying dendextend functions to apply to the row or column dendrogram. See details for usage.

split_rows, split_cols
: Vectors for splitting the rows and columns into facets. Can be a numeric vector shorter than the number of rows/columns to split the heatmap after those indices, or a vector of the same length as the number of rows/columns containing the facet memberships. In the latter case names can be used to match with rows/columns. Alternatively, if clustering is applied a single numeric value is accepted for the number of clusters to divide the plot into.

split_rows_side, split_cols_side
: Which side the row/column facet strips should be drawn on ('left'/'right', 'top'/'bottom').

Details

When using mixed layouts (layout is length two), mode needs to be length two as well, specifying the mode to use in each triangle. The cell_label_* and border_* arguments can all be length one to apply to the whole heatmap, length two vectors to apply to each triangle, or lists of length two, each element containing one value (apply to whole triangle) or a value per cell (apply cell-wise in triangle). cell_labels can also be specified per triangle, either as a logical vector of length two, or a list of length two containing a mix of logicals and matrices/data frames.

It is also possible to provide two scales for filling or colouring the triangles differently. In this case the col_scale must be one character value (scale used for both triangles) or NULL or a list of length two containing the scales to use (character or scale object, or NULL for default). size_scale works in the same way (but takes no character values). In addition, the scale-modifying arguments bins, na_col and limits can also be specified per triangle. limits must be a list of length two (or one) where each element is a numeric vector of length two.

The annotation parameter arguments annot_rows_params and annot_cols_params should be named lists, where the possible options correspond to the different annot_* arguments. The possible options are "dist" (distance between heatmap and annotation), "gap" (distance between annotations), "size" (cell size), "show_names" (logical, if the annotation names should be displayed), "border_col" (colour of border) and "border_lwd" (border line width). Any unused options will use the defaults set by the annot_* arguments.

The dendrogram parameters arguments dend_rows_params and dend_cols_params should be named lists, analogous to the annotation parameter arguments. Possible options are "col" (line colour), "dist" (distance from heatmap to dendrogram), "height" (height scaling), "lwd" (line width), and "lty" (line type).

The dend_rows_extend and dend_cols_extend arguments make it possible to customise the dendrograms using the dendextend package. The argument should be a named list, each element named after the dendextend function to use (consecutive usage of the set function is supported due to duplicate list names being possible). Each element should contain any arguments given to the dendextend function, such as the what argument used in the set function. Alternatively, dendextend functions can be provided in a functional sequence ("fseq" object) by piping together functions using the %>% pipe. Functions modifying the labels do not work as the dendrogram labels

are not displayed (they are in the axis text). As dendextend::as.ggdend() is used for conversion of the dendrogram, anything not supported by as.ggdend() will not work (such as "nodes_bg" or "rect.dendrogram"). See examples and the clustering article for example usage.

Value

The heatmap as a ggplot object. If return_data is TRUE the output is a list containing the plot (named 'plot'), the plotting data ('plot_data', with factor columns 'row' and 'col', a column 'value' containing the cell values, and 'layout' which part of the heatmap each cell belongs to), and the result of the clustering ('row_clustering' and/or 'col_clustering).

Examples

```
library(ggplot2)

# Use part of the mtcars data (for visibility)
hm_in <- mtcars[1:15, ]

# Basic usage
gghm(hm_in)

# Different layout (using a symmetric matrix)
gghm(cor(mtcars), layout = "tl")

# Mixed layouts
gghm(cor(mtcars), layout = c("tr", "bl"),
     # Hide one of the legends
     legend_order = c(1, NA))

# With clustering
gghm(scale(hm_in), cluster_rows = TRUE, cluster_cols = TRUE)

# Adjusting cluster dendrograms using common and specific options
gghm(scale(hm_in), cluster_rows = TRUE, cluster_cols = TRUE,
     # Common options
     dend_lwd = 0.7, dend_col = "magenta",
     # Specific options
     dend_rows_params = list(height = 1), dend_cols_params = list(lty = 2))

# With gaps
gghm(mtcars, split_rows = c(10, 20), split_cols = 6)

# With annotation and specifying colour scales
set.seed(123)
annot_rows <- data.frame(.names = rownames(hm_in),
                         annot1 = rnorm(nrow(hm_in)),
                         annot2 = sample(letters[1:3], nrow(hm_in), TRUE))
# Specify colour scale for one of the annotations (viridis mako)
annot_fill <- list(annot1 = "G")

gghm(scale(hm_in),
     # Change colours of heatmap (Brewer Purples)
```

```
    col_scale = "Purples",
    annot_rows_df = annot_rows, annot_rows_col = annot_fill) +
    # Use ggplot2::theme to adjust margins to fit the annotation names
    theme(plot.margin = margin(30, 10, 60, 20))

# Using the dend_*_extend arguments
gghm(scale(hm_in), cluster_rows = TRUE, dend_rows_extend =
  list("set" = list("branches_lty", c(1, 2, 3)),
       # Empty list element (or NULL) if no arguments to be given
       "highlight_branches_col" = list()))
```

---

gghm_tidy `gghm()` *for long format data.*

Description

`gghm()` for long format data.

Usage

```
gghm_tidy(
  x,
  rows,
  cols,
  values,
  labels = NULL,
  annot_rows = NULL,
  annot_cols = NULL,
  facet_rows = NULL,
  facet_cols = NULL,
  ...
)
```

Arguments

`x`	Data frame containing data to plot.
`rows`, `cols`, `values`	Columns to use as rows, columns, and cell values.
`labels`	Column to use for cell labels. NULL (default) for no labels.
`annot_rows`, `annot_cols`	Columns to use for row and column annotations.
`facet_rows`, `facet_cols`	Columns to use for row/column facet memberships.
`...`	Additional arguments for `gghm()`.

Value

A ggplot2 object with the heatmap. If `return_data` is `TRUE`, plotting data is returned as well.

Examples

```
# Basic example
set.seed(123)
hm_in <- data.frame(row = rep(letters[1:10], each = 5),
                    col = rep(LETTERS[1:5], 10),
                    val = rnorm(50))
gghm_tidy(hm_in, row, col, val)

# Annotation and clustering
# Add annotation by giving names of columns in the data
hm_in$row_annot1 <- rep(1:10, each = 5)
hm_in$row_annot2 <- rep(10:1, each = 5)
hm_in$col_annot <- rep(letters[1:5], 10)
# Columns are given using 'tidy' selection
# so they can be unquoted, quoted, from variables (with !! notation) or indices
gghm_tidy(hm_in, row, col, val,
          annot_rows = c(row_annot1, row_annot2),
          annot_cols = col_annot,
          cluster_rows = TRUE,
          cluster_cols = TRUE)

# Add cell labels
hm_in$lab <- 1:50
gghm_tidy(hm_in, row, col, val,
          labels = lab, cell_label_col = "white")
```

Index