Package 'plotthis'

June 10, 2026

Title High-Level Plotting Built Upon 'ggplot2' and Other Plotting Packages

Version 0.12.1

Description Provides high-level API and a wide range of options to create stunning, publication-quality plots effortlessly.
It is built upon 'ggplot2' and other plotting packages, and is designed to be easy to use and to work seamlessly with 'ggplot2' objects.
It is particularly useful for creating complex plots with multiple layers, facets, and annotations.
It also provides a set of functions to create plots for specific types of data, such as Venn diagrams, alluvial diagrams, and phylogenetic trees.
The package is designed to be flexible and customizable, and to work well with the 'ggplot2' ecosystem.
The API can be found at <https://pwwang.github.io/plotthis/reference/index.html>.

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/pwwang/plotthis
https://pwwang.github.io/plotthis/

BugReports https://github.com/pwwang/plotthis/issues

RoxygenNote 7.3.3

Depends R (>= 4.2.0)

Imports circlize, ggplot2, rlang, dplyr, tidyr, glue, forcats, gtable, reshape2, stringr, scales, gridtext, methods, patchwork, ggrepel, ggnewscale, cowplot, zoo

Suggests plotly, testthat, Matrix, alluvial, datasets, ComplexHeatmap, cluster, clustree, gglogger, ggwordcloud, ggalluvial, ggVennDiagram (>= 1.5.0), ggupset, ggpubr, ggbeeswarm, ggforce, ggraph, ggridges, ggmanh, qqplotr, hexbin, igraph, iNEXT, scattermore, sf, terra, concaveman, plotROC, OptimalCutpoints, proxyC, metR

LazyData true

Additional_repositories https://bioconductor.org/packages/release/bioc

Config/Needs/website rmarkdown

NeedsCompilation no

Author Panwen Wang [aut, cre] (ORCID: <https://orcid.org/0000-0002-4614-8970>)

Maintainer Panwen Wang <pwwang@pwwang.com>

Repository CRAN

Date/Publication 2026-06-10 05:20:19 UTC

Contents

`AreaPlot` *Area plot*

Description

A plot showing how one or more groups' numeric values change over the progression of a another variable

Usage

```
AreaPlot(
  data,
  x,
  y = NULL,
  x_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  scale_y = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
```

```
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of fill.
scale_y	A logical value to scale the y-axis by the total number in each x-axis group.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.

alpha
: A numeric value specifying the transparency of the plot.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

x_text_angle
: A numeric value specifying the angle of the x-axis text.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

  - `FALSE` (default): Drop empty factor levels from the data before plotting.
  - `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
  - `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
axes	A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axes in individual plots. • 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout. • 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.
axis_titles	A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axis titles in individual plots. • 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction. • 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.
guides	A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'collect' will collect guides below to the given nesting level, removing duplicates. • 'keep' will stop collection at this level and let guides be placed alongside their plot. • 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.
design	Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
data <- data.frame(
    x = rep(c("A", "B", "C", "D"), 2),
    y = c(1, 3, 6, 4, 2, 5, 7, 8),
    group = rep(c("F1", "F2"), each = 4),
    split = rep(c("X", "Y"), 4)
```

```
)
AreaPlot(data, x = "x", y = "y", group_by = "group")
AreaPlot(data, x = "x", y = "y", group_by = "group",
         scale_y = TRUE)
AreaPlot(data, x = "x", y = "y", split_by = "group")
AreaPlot(data, x = "x", y = "y", split_by = "group", palette = c(F1 = "Blues", F2 = "Reds"))
AreaPlot(data, x = "x", y = "y", group_by = "group", split_by = "split",
    legend.direction = c(X = "horizontal", Y = "vertical"),
    legend.position = c(X = "top", Y = "right"))

# How keep_na and keep_empty work
data <- data.frame(
    x = factor(rep(c("A", NA, "C", "D"), 3), levels = c("A", "B", "C", "D")),
    y = c(1, 3, 6, 4, 2, 5, 7, 8, 4, 2, 3, 5),
   group = factor(sample(rep(c("F1", NA, "F3"), each = 4)), levels = c("F1", "F2", "F3")),
    split = factor(sample(rep(c("X", "Y", NA), 4)), levels = c("X", "Y", "Z")),
    facet = factor(sample(rep(c("M", "N", NA), 4)), levels = c("M", "N", "O"))
)

AreaPlot(data, x = "x", y = "y", group_by = "group")
AreaPlot(data, x = "x", y = "y", group_by = "group", keep_na = TRUE, keep_empty = TRUE)
AreaPlot(data, x = "x", y = "y", group_by = "group", keep_na = TRUE, keep_empty = "level")
AreaPlot(data, x = "x", y = "y", group_by = "group", keep_na = FALSE, keep_empty = TRUE)
AreaPlot(data, x = "x", y = "y", group_by = "group",
    keep_na = list(x = TRUE, group = FALSE), keep_empty = list(x = FALSE, group = TRUE))
AreaPlot(data, x = "x", y = "y", group_by = "group",
    keep_na = list(x = FALSE, group = TRUE), keep_empty = list(x = TRUE, group = FALSE))
```

BarPlot *Bar Plot*

Description

- BarPlot is used to create a bar plot.
- SplitBarPlot (a.k.a WaterfallPlot) is used to create a bar plot with splitting the bars on the two sides.

Usage

```
BarPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  flip = FALSE,
  fill_by = TRUE,
  fill_name = NULL,
  line_name = NULL,
```

```
  label_nudge = 0.02,
  label = NULL,
  label_fg = "black",
  label_size = 4,
  label_bg = "white",
  label_bg_r = 0.1,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  facet_args = list(),
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_line = NULL,
  line_color = "red2",
  line_width = 0.6,
  line_type = 2,
  add_trend = FALSE,
  trend_color = "black",
  trend_linewidth = 1,
  trend_ptsize = 2,
  theme = "theme_this",
  theme_args = list(),
  palette = NULL,
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  x_text_angle = 0,
  aspect.ratio = 1,
  y_min = NULL,
  y_max = NULL,
  position = "auto",
  position_dodge_preserve = "total",
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
```

```
  keep_na = FALSE,
  expand = waiver(),
  width = waiver(),
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

SplitBarPlot(
  data,
  x,
  y,
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  alpha_by = NULL,
  alpha_reverse = FALSE,
  alpha_name = NULL,
 order_y = list(`+` = c("x_desc", "alpha_desc"), `-` = c("x_desc", "alpha_asc")),
  bar_height = 0.9,
  lineheight = 0.5,
  max_charwidth = 80,
  fill_by = NULL,
  fill_by_sep = "_",
  fill_name = NULL,
  direction_name = "direction",
  direction_pos_name = "positive",
  direction_neg_name = "negative",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  facet_by = NULL,
  facet_scales = "free_y",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  x_min = NULL,
```

```
  x_max = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  keep_na = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

WaterfallPlot(
  data,
  x,
  y,
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  alpha_by = NULL,
  alpha_reverse = FALSE,
  alpha_name = NULL,
 order_y = list(`+` = c("x_desc", "alpha_desc"), `-` = c("x_desc", "alpha_asc")),
  bar_height = 0.9,
  lineheight = 0.5,
  max_charwidth = 80,
  fill_by = NULL,
  fill_by_sep = "_",
  fill_name = NULL,
  direction_name = "direction",
  direction_pos_name = "positive",
  direction_neg_name = "negative",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  facet_by = NULL,
```

```
  facet_scales = "free_y",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  x_min = NULL,
  x_max = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_empty = FALSE,
  keep_na = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
y	A character string specifying the column name of the data frame to plot for the y-axis.
flip	A logical value indicating whether to flip the x and y axes.
fill_by	A variable used to fill the bars. Both character/factor and numeric columns are accepted. If TRUE (default), the bars will be filled by the x-axis values, If FALSE, the bars will be filled a single color (the first color in the palette). ONLY works when group_by is NULL. When group_by is not NULL, the bars will be filled by the group_by variable.
fill_name	A character string to specify the name of the fill variable in the legend. Only works when fill_by applies.
line_name	A character string indicating the name of the line.
label_nudge	A numeric value to nudge the labels (the distance between the label and the top of the bar).

label	A column name for the values to be displayed on the top of the bars. If TRUE, the y values will be displayed.
label_fg	A character string indicating the color of the label.
label_size	A numeric value indicating the size of the label.
label_bg	A character string indicating the background color of the label.
label_bg_r	A numeric value indicating the radius of the background.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to specify the name of the group_by in the legend.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
facet_args	A list of arguments to pass to ggplot2::facet_grid or ggplot2::facet_wrap.
add_bg	A logical value indicating whether to add a background to the plot.
bg_palette	A character string indicating the palette to use for the background.
bg_palcolor	A character string indicating the color to use for the background.
bg_alpha	A numeric value indicating the alpha of the background.
add_line	A numeric value indicating the y value to add a horizontal line.
line_color	A character string indicating the color of the line.
line_width	A numeric value indicating the size of the line.
line_type	A numeric value indicating the type of the line.
add_trend	A logical value to add trend line to the plot.
trend_color	A character string to specify the color of the trend line.
trend_linewidth	A numeric value to specify the width of the trend line.
trend_ptsize	A numeric value to specify the size of the trend line points.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use for the bars. When group_by is specified, it defaults to "Paired". When group_by is not specified, it defaults to "Spectral" if fill_by column is numeric, otherwise "Paired".

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

alpha
: A numeric value specifying the transparency of the plot.

x_text_angle
: A numeric value specifying the angle of the x-axis text.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

y_min
: A numeric value to specify the minimum value of the y axis.

y_max
: A numeric value to specify the maximum value of the y axis.

position
: A character string indicating the position of the bars. If "auto", the position will be "stack" if group_by has more than 5 levels, otherwise "dodge". "fill" is also a valid option. Only works when group_by is not NULL.

position_dodge_preserve
: Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

  - FALSE (default): Drop empty factor levels from the data before plotting.
  - TRUE: Keep empty factor levels and show them as a separate category in the plot.
  - "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed

from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

`expand` The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

`width` A numeric value specifying the width of the bars.

`combine` Whether to combine the plots into one when facet is FALSE. Default is TRUE.

`nrow` A numeric value specifying the number of rows in the facet.

`ncol` A numeric value specifying the number of columns in the facet.

`byrow` A logical value indicating whether to fill the plots by row.

`seed` The random seed to use. Default is 8525.

`axes` A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.

- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

`y_sep` A character string to concatenate the x columns if there are multiple.

`alpha_by` A character string indicating the column name to use for the transparency of the bars.

`alpha_reverse` A logical value indicating whether to reverse the transparency.

`alpha_name` A character string indicating the legend name of the transparency.

`order_y` A list of character strings indicating the order of the y axis. The keys are "+", "-", or "*". However, "+/-" should not be mixed with "*". The values are "x_asc", "x_desc", "alpha_asc", or "alpha_desc", indicating how to order the y axis. The default is `list("+" = c("x_desc", "alpha_desc"), "-" = c("x_desc", "alpha_asc"))`, meaning the positive values are ordered by the x-axis values in descending order and the alpha values in descending order, and the negative values are ordered by the x-axis values in descending order and the alpha values in ascending order. The "*" key is used to order the y axis without considering the direction.

`bar_height` A numeric value indicating the height of the bars.

`lineheight` A numeric value indicating the height of the text.

`max_charwidth` A numeric value indicating the maximum width of the text.

`fill_by_sep` A character string to concatenate the fill columns if there are multiple.

`direction_name` A character string indicating the name of the direction column and will be shown in the legend.

`direction_pos_name` A character string indicating the name of the positive direction.

`direction_neg_name` A character string indicating the name of the negative direction.

`x_min` A numeric value indicating the minimum value of the x axis.

`x_max` A numeric value indicating the maximum value of the x axis.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
    x = c("A", "B", "C", "D", "E", "F", "G", "H"),
    y = c(10, 8, 16, 4, 6, 12, 14, 2),
    group = c("G1", "G1", "G2", "G2", "G3", "G3", "G4", "G4"),
    facet = c("F1", "F2", "F3", "F4", "F1", "F2", "F3", "F4")
```

```
)

BarPlot(data, x = "x", y = "y")
BarPlot(data, x = "x", y = "y", fill_by = FALSE)
BarPlot(data, x = "x", y = "y", label = TRUE)
BarPlot(data, x = "x", y = "y", label = "facet", label_nudge = 0)
BarPlot(data, x = "group", y = "y", group_by = "x")
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    position = "dodge", add_bg = TRUE
)
BarPlot(data,
    x = "x", y = "y", split_by = "group",
    facet_by = "facet", position = "dodge", facet_ncol = 1
)
BarPlot(data,
    x = "x", y = "y", split_by = "group", facet_by = "facet",
    position = "dodge", facet_ncol = 1, guides = 'collect'
)
BarPlot(data,
    x = "x", y = "y", split_by = "group",
    palette = list(G1 = "Reds", G2 = "Blues", G3 = "Greens", G4 = "Purp"),
    facet_by = "facet", position = "dodge", facet_ncol = 1
)
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    position = "dodge", add_bg = TRUE, bg_palette = "Spectral"
)
# use the count
BarPlot(data, x = "group", ylab = "count")
# flip the plot
BarPlot(data, x = "group", flip = TRUE, ylab = "count")
# Allow numeric fill_by
BarPlot(data, x = "x", y = "y", fill_by = "y", flip = TRUE)

data <- data.frame(
    x = factor(c("A", "B", "C", "D", "E", "F", NA, "H"), levels = LETTERS[1:10]),
    y = c(10, 8, 16, 4, 6, NA, 14, 2),
    group = factor(c("G1", "G1", "G2", NA, "G3", "G3", "G5", "G5"),
       levels = c("G1", "G2", "G3", "G4", "G5")),
    facet = factor(c("F1", NA, "F3", "F4", "F1", "F2", "F3", "F4"),
       levels = c("F1", "F2", "F3", "F4", "F5"))
)
# keep_na and keep_empty on simple barplots
BarPlot(data, x = "x", y = "y",
    title = "keep_na = FALSE; keep_empty = FALSE")
BarPlot(data, x = "x", y = "y",
    keep_na = TRUE, keep_empty = TRUE,
    title = "keep_na = TRUE; keep_empty = TRUE")
BarPlot(data, x = "x", y = "y",
    keep_na = TRUE, keep_empty = TRUE, facet_by = "facet",
    title = "keep_na = TRUE; keep_empty = TRUE; facet_by = 'facet'")
BarPlot(data, x = "x", y = "y",
```

```
    keep_na = TRUE, keep_empty = 'level',
    title = "keep_na = TRUE; keep_empty = 'level'")
BarPlot(data, x = "x", y = "y",
    keep_na = list(x = TRUE), keep_empty = list(x = FALSE),
    title = "keep_na: x=TRUE\nkeep_empty: x=FALSE"
)

# keep_na and keep_empty on grouped barplots
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    title = "keep_na = FALSE; keep_empty = FALSE")
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    keep_na = TRUE, keep_empty = TRUE,
    title = "keep_na = TRUE; keep_empty = TRUE"
)
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    keep_na = TRUE, keep_empty = TRUE, facet_by = "facet",
    title = "keep_na = TRUE; keep_empty = TRUE; facet_by = 'facet'"
)
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    keep_na = TRUE, keep_empty = 'level',
    title = "keep_na = TRUE; keep_empty = 'level'"
)
BarPlot(data,
    x = "group", y = "y", group_by = "x",
    keep_na = list(x = TRUE, group = FALSE),
    keep_empty = list(x = FALSE, group = TRUE),
    title = "keep_na: x=TRUE, group=FALSE\nkeep_empty: x=FALSE, group=TRUE"
)


set.seed(8525)
data <- data.frame(
    word = c("apple", "banana", "cherry", "date", "elderberry",
             "It is a very long term with a lot of words"),
    count = c(-10, 20, -30, 40, 50, 34),
    score = c(1, 2, 3, 4, 5, 3.2),
    group = c("A", "A", "B", "B", "C", "C")
)
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score")
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
             max_charwidth = 30, lineheight = 1.1)
SplitBarPlot(data, x = "count", y = "word", fill_by = "group")
SplitBarPlot(data, x = "count", y = "word", facet_by = "group",
             fill_name = "Direction")
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score", split_by="group",
             palette = c(A = "Reds", B = "Blues", C = "Greens"))

# test keep_na and keep_empty
data <- data.frame(
```

```
    word = factor(c("apple", "banana", "cherry", NA, "elderberry",
         "It is a very long term with a lot of words"),
         levels = c("apple", "banana", "cherry", "date", "elderberry",
             "unused", "It is a very long term with a lot of words")),
    count = c(-10, 20, NA, 40, 10, 34),
    score = c(1, 2, 3, 4, 5, 3.2),
    group = factor(sample(c("A", "A", "B", "B", "C", "C")),
         levels = c("A", "B", "C", "D"))
)

SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
    title = "keep_na = FALSE; keep_empty = FALSE")
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
    keep_na = TRUE, keep_empty = TRUE,
    title = "keep_na = TRUE; keep_empty = TRUE")
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
    keep_na = TRUE, keep_empty = TRUE, facet_by = "group",
    title = "keep_na = TRUE; keep_empty = TRUE; facet_by = 'group'")
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
    keep_na = TRUE, keep_empty = "level",
    title = "keep_na = TRUE; keep_empty = 'level'")
SplitBarPlot(data, x = "count", y = "word", alpha_by = "score",
    keep_na = list(word = FALSE), keep_empty = list(word = TRUE),
    title = "keep_na: word=FALSE; keep_empty: word=TRUE")
```

BoxPlot *Box / Violin / Bar Plot*

Description

Box plot, bar plot (mean values), or violin plot with optional jitter points, trend line, statistical test, background, line, and highlight. When `base = "bar"`, bars show the mean values with optional error bars (SEM, SD, or CI).

Usage

```
BoxPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  base = c("box", "bar"),
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  symnum_args = NULL,
  sort_x = NULL,
  flip = FALSE,
```

```
    keep_empty = FALSE,
    keep_na = FALSE,
    group_by = NULL,
    group_by_sep = "_",
    group_name = NULL,
    paired_by = NULL,
    x_text_angle = ifelse(isTRUE(flip), 0, 45),
    step_increase = 0.1,
    fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
    palreverse = FALSE,
    position_dodge_preserve = "total",
    theme = "theme_this",
    theme_args = list(),
    palette = "Paired",
    palcolor = NULL,
    alpha = 1,
    aspect.ratio = NULL,
    legend.position = "right",
    legend.direction = "vertical",
    add_point = FALSE,
    pt_color = if (isTRUE(add_beeswarm)) NULL else "grey30",
    pt_size = NULL,
    pt_alpha = 1,
    jitter_width = NULL,
    jitter_height = 0,
    stack = FALSE,
    y_max = NULL,
    y_min = NULL,
    add_beeswarm = FALSE,
    beeswarm_method = "swarm",
    beeswarm_cex = 1,
    beeswarm_priority = "ascending",
    beeswarm_dodge = 0.9,
    add_trend = FALSE,
    trend_color = NULL,
    trend_linewidth = 1,
    trend_ptsize = 2,
    add_stat = NULL,
    stat_name = NULL,
    stat_color = "black",
    stat_size = 1,
    stat_stroke = 1,
    stat_shape = 25,
    add_errorbar = "SEM",
    errorbar_color = "grey20",
    errorbar_width = 0.4,
    errorbar_linewidth = 0.6,
    add_bg = FALSE,
```

```
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_line = NULL,
  line_color = "red2",
  line_width = 0.6,
  line_type = 2,
  highlight = NULL,
  highlight_color = "red2",
  highlight_size = 1,
  highlight_alpha = 1,
  comparisons = NULL,
  ref_group = NULL,
  pairwise_method = "wilcox.test",
  multiplegroup_comparisons = FALSE,
  multiple_method = "kruskal.test",
  sig_label = "p.format",
  sig_labelsize = 3.5,
  hide_ns = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  ...
)

ViolinPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  symnum_args = NULL,
```

```
  sort_x = NULL,
  flip = FALSE,
  keep_empty = FALSE,
  keep_na = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  paired_by = NULL,
  x_text_angle = ifelse(isTRUE(flip), 0, 45),
  step_increase = 0.1,
  fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
  palreverse = FALSE,
  position_dodge_preserve = "total",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  add_point = FALSE,
  pt_color = if (isTRUE(add_beeswarm)) NULL else "grey30",
  pt_size = NULL,
  pt_alpha = 1,
  jitter_width = NULL,
  jitter_height = 0,
  stack = FALSE,
  y_max = NULL,
  y_min = NULL,
  add_beeswarm = FALSE,
  beeswarm_method = "swarm",
  beeswarm_cex = 1,
  beeswarm_priority = "ascending",
  beeswarm_dodge = 0.9,
  add_box = FALSE,
  box_color = "black",
  box_width = 0.1,
  box_ptsize = 2.5,
  add_trend = FALSE,
  trend_color = NULL,
  trend_linewidth = 1,
  trend_ptsize = 2,
  add_stat = NULL,
  stat_name = NULL,
  stat_color = "black",
  stat_size = 1,
  stat_stroke = 1,
```

```
  stat_shape = 25,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_line = NULL,
  line_color = "red2",
  line_width = 0.6,
  line_type = 2,
  highlight = NULL,
  highlight_color = "red2",
  highlight_size = 1,
  highlight_alpha = 1,
  comparisons = NULL,
  ref_group = NULL,
  pairwise_method = "wilcox.test",
  multiplegroup_comparisons = FALSE,
  multiple_method = "kruskal.test",
  sig_label = "p.format",
  sig_labelsize = 3.5,
  hide_ns = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  ...
)

BeeswarmPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
```

```
  split_by_sep = "_",
  symnum_args = NULL,
  sort_x = NULL,
  flip = FALSE,
  keep_empty = FALSE,
  keep_na = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  paired_by = NULL,
  x_text_angle = ifelse(isTRUE(flip), 0, 45),
  step_increase = 0.1,
  fill_mode = ifelse(!is.null(group_by), "dodge", "x"),
  palreverse = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  pt_color = NULL,
  pt_size = NULL,
  pt_alpha = 1,
  position_dodge_preserve = "total",
  jitter_width = NULL,
  jitter_height = 0,
  stack = FALSE,
  y_max = NULL,
  y_min = NULL,
  add_violin = FALSE,
  beeswarm_method = "swarm",
  beeswarm_cex = 1,
  beeswarm_priority = "ascending",
  beeswarm_dodge = 0.9,
  add_box = FALSE,
  box_color = "black",
  box_width = 0.1,
  box_ptsize = 2.5,
  add_trend = FALSE,
  trend_color = NULL,
  trend_linewidth = 1,
  trend_ptsize = 2,
  add_stat = NULL,
  stat_name = NULL,
  stat_color = "black",
  stat_size = 1,
```

```
  stat_stroke = 1,
  stat_shape = 25,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_line = NULL,
  line_color = "red2",
  line_width = 0.6,
  line_type = 2,
  highlight = NULL,
  highlight_color = "red2",
  highlight_size = 1,
  highlight_alpha = 1,
  comparisons = NULL,
  ref_group = NULL,
  pairwise_method = "wilcox.test",
  multiplegroup_comparisons = FALSE,
  multiple_method = "kruskal.test",
  sig_label = "p.format",
  sig_labelsize = 3.5,
  hide_ns = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are pro-

vided. When in_form is "wide", x columns will not be concatenated.

y — A character string specifying the column name of the data frame to plot for the y-axis.

base — A character string to specify the base plot type. Either "box", "violin", "bar" or "none" (used by BeeswarmPlot). When "bar", bars showing the mean values are plotted. This is mutually exclusive with add_box.

in_form — A character string to specify the input data type. Either "long" or "wide".

split_by — The column(s) to split data by and plot separately.

split_by_sep — The separator for multiple split_by columns. See split_by

symnum_args — A list of arguments to pass to the function symnum for symbolic number coding of p-values. For example, symnum_args <- list(cutpoints = c(0, 0.0001, 0.001, 0.01, 0.05, Inf), symbols = c("****", "***", "**", "*", "ns")). In other words, we use the following convention for symbols indicating statistical significance:

- ns: p > 0.05
- *: p <= 0.05
- **: p <= 0.01
- ***: p <= 0.001
- ****: p <= 0.0001

sort_x — An expression (in character string) to order x-axis. For example, "mean(y)" will order the x-axis by the mean of y. Default is NULL, which means keeping the original order of x. Note that when keep_empty is TRUE for x, the empty x levels will always be placed at the end of the x-axis.

flip — A logical value to flip the plot.

keep_empty — One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

keep_na — A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of dodge.
paired_by	A character string of the column name identifying paired observations for paired tests.
x_text_angle	A numeric value specifying the angle of the x-axis text.
step_increase	A numeric value to specify the step increase in fraction of total height for every additional comparison of the significance labels.
fill_mode	A character string to specify the fill mode. Either "dodge", "x", "mean", "median".
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
position_dodge_preserve	Should dodging preserve the "total" width of all elements at a position, or the width of a "single" element?
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
alpha	A numeric value specifying the transparency of the plot.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
add_point	A logical value to add (jitter) points to the plot.
pt_color	A character string to specify the color of the points.
pt_size	A numeric value to specify the size of the points.
pt_alpha	A numeric value to specify the transparency of the points.
jitter_width	A numeric value to specify the width of the jitter. Defaults to 0.5, but when paired_by is provided, it will be set to 0.
jitter_height	A numeric value to specify the height of the jitter.
stack	A logical value whether to stack the facetted plot by 'facet_by'.

y_max A numeric value or a character string to specify the maximum value of the y-axis. You can also use quantile notation like "q95" to specify the 95th percentile. When comparisons are set and a numeric y_max is provided, it will be used to set the y-axis limit, including the significance labels.

y_min A numeric value or a character string to specify the minimum value of the y-axis. You can also use quantile notation like "q5" to specify the 5th percentile.

add_beeswarm A logical value to add beeswarm points to the plot instead of jittered points. When TRUE, points are positioned using the beeswarm algorithm to avoid overlap while showing density. Requires the ggbeeswarm package to be installed.

beeswarm_method A character string to specify the beeswarm method. Either "swarm", "compactswarm", "hex", "square", or "center". Default is "swarm". See ggbeeswarm::geom_beeswarm for details.

beeswarm_cex A numeric value to specify the scaling for adjusting point spacing in beeswarm. Default is 1. Larger values space out points more.

beeswarm_priority A character string to specify point layout priority. Either "ascending", "descending", "density", or "random". Default is "ascending".

beeswarm_dodge A numeric value to specify the dodge width for beeswarm points when group_by is provided. Default is 0.9

add_trend A logical value to add trend line to the plot.

trend_color A character string to specify the color of the trend line. This won't work when group_by is specified, the trend line will be colored by the group_by variable.#'

trend_linewidth A numeric value to specify the width of the trend line.

trend_ptsize A numeric value to specify the size of the trend line points.

add_stat A character string to add statistical test to the plot.

stat_name A character string to specify the name of the stat legend.

stat_color A character string to specify the color of the statistical test.

stat_size A numeric value to specify the size of the statistical test.

stat_stroke A numeric value to specify the stroke of the statistical test.

stat_shape A numeric value to specify the shape of the statistical test.

add_errorbar A character string to specify the type of error bars to add to bar plots. Only available when base = "bar". Case insensitive. Available options are:

- "SEM" (default): Standard error of the mean.
- "SD": Standard deviation.
- "CI" or "CIXX" (e.g., "CI95"): Confidence interval. "CI" defaults to "CI95" (95\
- "none": No error bars.

errorbar_color A character string to specify the color of the error bars. Default is "black".

errorbar_width A numeric value to specify the width of the error bar caps. Default is 0.5.

errorbar_linewidth
: A numeric value to specify the line width of the error bars. Default is 0.75.

add_bg
: A logical value to add background to the plot.

bg_palette
: A character string to specify the palette of the background.

bg_palcolor
: A character vector to specify the colors of the background.

bg_alpha
: A numeric value to specify the transparency of the background.

add_line
: A character string to add a line to the plot.

line_color
: A character string to specify the color of the line.

line_width
: A numeric value to specify the size of the line.

line_type
: A numeric value to specify the type of the line.

highlight
: A vector of character strings to highlight the points. It should be a subset of the row names of the data. If TRUE, it will highlight all points.

highlight_color
: A character string to specify the color of the highlighted points.

highlight_size
: A numeric value to specify the size of the highlighted points.

highlight_alpha
: A numeric value to specify the transparency of the highlighted points.

comparisons
: A logical value or a list of vectors to perform pairwise comparisons. If `TRUE`, it will perform pairwise comparisons for all pairs.

ref_group
: A character string to specify the reference group for comparisons.

pairwise_method
: A character string to specify the pairwise comparison method.

multiplegroup_comparisons
: A logical value to perform multiple group comparisons.

multiple_method
: A character string to specify the multiple group comparison method.

sig_label
: A character string to specify the label of the significance test. For multiple group comparisons (`multiplegroup_comparisons = TRUE`), it must be either "p.format" or "p.signif". For pairwise comparisons, it can be:

  - the column containing the label (e.g.: label = "p" or label = "p.adj"), where p is the p-value. Other possible values are "p.signif", "p.adj.signif", "p.format", "p.adj.format".
  - an expression that can be formatted by the glue() package. For example, when specifying `label = "Wilcoxon, p = {p}"`, the expression `{p}` will be replaced by its value.
  - a combination of plotmath expressions and glue expressions. You may want some of the statistical parameter in italic; for example: `label = "Wilcoxon, p= {p}"` See https://rpkgs.datanovia.com/ggpubr/reference/geom_pwc.html for more details.

sig_labelsize
: A numeric value to specify the size of the significance test label.

hide_ns
: A logical value to hide the non-significant comparisons.

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

axes
: A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
  - 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides
: A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'collect' will collect guides below to the given nesting level, removing duplicates.
  - 'keep' will stop collection at this level and let guides be placed alongside their plot.

- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

...	Additional arguments.
add_box	A logical value to add box plot to the plot.
box_color	A character string to specify the color of the box plot.
box_width	A numeric value to specify the width of the box plot.
box_ptsize	A numeric value to specify the size of the box plot points in the middle.
add_violin	Logical, whether to add violin plot behind the beeswarm points. Adding violin to a beeswarm plot is actually not supported. A message will be shown to remind users to use `ViolinPlot(..., add_beeswarm = TRUE)` instead.

Value

The Box / Violin plot(s). When `split_by` is not provided, it returns a ggplot object. When `split_by` is provided, it returns a object of plots wrapped by `patchwork::wrap_plots` if `combine = TRUE`; otherwise, it returns a list of ggplot objects.

Examples

```
set.seed(8525)
data <- data.frame(
    x = rep(LETTERS[1:8], each = 40),
    y = c(rnorm(160), rnorm(160, mean = 1)),
    group1 = sample(c("g1", "g2"), 320, replace = TRUE),
    group2 = sample(c("h1", "h2", "h3", "h4"), 320, replace = TRUE)
)

BoxPlot(data, x = "x", y = "y")
BoxPlot(data, x = "x", y = "y", add_beeswarm = TRUE, pt_color = "grey30")
BoxPlot(data,
    x = "x", y = "y",
    stack = TRUE, flip = TRUE, facet_by = "group1",
    add_bg = TRUE, bg_palette = "Paired"
)
BoxPlot(data,
    x = "x", y = "y",
    stack = TRUE, flip = TRUE, split_by = "group1",
    add_bg = TRUE, bg_palette = "Paired",
    palcolor = list(g1 = c("red", "blue"), g2 = c("blue", "red"))
)

# sort_x
data <- data.frame(
  x = factor(rep(LETTERS[1:5], each = 40),
     levels = c(LETTERS[1:2], "unused", LETTERS[3:5])),
  y = c(rnorm(40, mean = 5), rnorm(40, mean = 4), rnorm(40, mean = 3),
     rnorm(40, mean = 2), rnorm(40, mean = 1))
)
BoxPlot(data, x = "x", y = "y", sort_x = "mean(y)", keep_empty = TRUE)
BoxPlot(data, x = "x", y = "y", sort_x = "mean(-y)", keep_empty = TRUE)
```

```
# wide form data
data_wide <- data.frame(
    A = rnorm(100),
    B = rnorm(100),
    C = rnorm(100)
)
BoxPlot(data_wide, x = c("A", "B", "C"), in_form = "wide")

paired_data <- data.frame(
    subject = rep(paste0("s", 1:10), each = 2),
    visit = rep(c("pre", "post"), times = 10),
    value = rnorm(20)
)
# paired plot with connected lines and paired test
BoxPlot(
    paired_data,
    x = "visit", y = "value", comparisons = TRUE,
    paired_by = "subject", add_point = TRUE
)
paired_group_data <- data.frame(
    subject = rep(paste0("s", 1:6), each = 2),
    x = rep(c("A", "B"), each = 6),
    group = rep(c("before", "after"), times = 6),
    value = rnorm(12)
)
BoxPlot(
    paired_group_data,
    x = "x", y = "value",
    paired_by = "subject", group_by = "group",
    comparisons = TRUE, pt_size = 3, pt_color = "red"
)

# keep_na and keep_empty example
data <- data.frame(
    x = factor(rep(c(LETTERS[1:3], NA, LETTERS[5:8]), each = 40),
       levels = c(LETTERS[1:8])),
    y = c(rnorm(160), rnorm(160, mean = 1)),
    group1 = sample(c("g1", "g2"), 320, replace = TRUE),
    group2 = factor(sample(c("h1", NA, "h3", "h4"), 320, replace = TRUE),
       levels = c("h1", "h2", "h3", "h4"))
)

BoxPlot(data, x = "x", y = "y",
    title = "keep_na = FALSE; keep_empty = FALSE")
BoxPlot(data, x = "x", y = "y", keep_na = TRUE, keep_empty = TRUE,
    title = "keep_na = TRUE; keep_empty = TRUE")
BoxPlot(data, x = "x", y = "y", keep_na = TRUE, keep_empty = TRUE,
    title = "keep_na = TRUE; keep_empty = TRUE", facet_by = "group2")
BoxPlot(data, x = "x", y = "y", keep_na = TRUE, keep_empty = 'level',
    title = "keep_na = TRUE; keep_empty = 'level'")
BoxPlot(data, x = "x", y = "y", group_by = "group2",
    title = "keep_na = FALSE; keep_empty = FALSE; group_by = 'group2'")
```

```
BoxPlot(data, x = "x", y = "y", group_by = "group2",
    keep_na = TRUE, keep_empty = TRUE,
    title = "keep_na = TRUE; keep_empty = TRUE; group_by = 'group2'")
BoxPlot(data, x = "x", y = "y", group_by = "group2",
    keep_na = TRUE, keep_empty = 'level',
    title = "keep_na = TRUE; keep_empty = 'level'; group_by = 'group2'")
BoxPlot(data, x = "x", y = "y", group_by = "group2",
    keep_na = list(x = TRUE, group2 = FALSE),
    keep_empty = list(x = FALSE, group2 = TRUE),
    title = "keep_na: x=TRUE, group2=FALSE\nkeep_empty: x=FALSE, group2=TRUE"
)
BoxPlot(data, x = "x", y = "y", group_by = "group2",
    keep_na = list(x = FALSE, group2 = TRUE),
    keep_empty = list(x = TRUE, group2 = FALSE),
    title = "keep_na: x=FALSE, group2=TRUE\nkeep_empty: x=TRUE, group2=FALSE"
)

# Bar plot (base = "bar") shows mean values with error bars
data$y <- abs(data$y)  # make y values positive for better bar plot visualization
BoxPlot(data, x = "x", y = "y", base = "bar")
BoxPlot(data, x = "x", y = "y", base = "bar", add_errorbar = "SD")
BoxPlot(data, x = "x", y = "y", base = "bar", add_errorbar = "CI95")
BoxPlot(data, x = "x", y = "y", base = "bar", add_errorbar = "none")
BoxPlot(data, x = "x", y = "y", base = "bar", group_by = "group1")
BoxPlot(data, x = "x", y = "y", base = "bar", add_point = TRUE)
BoxPlot(data, x = "x", y = "y", base = "bar",
    fill_mode = "mean", palette = "Blues")


ViolinPlot(data, x = "x", y = "y")
ViolinPlot(data, x = "x", y = "y", add_beeswarm = TRUE, pt_color = "grey30")
ViolinPlot(data, x = "x", y = "y", add_box = TRUE)
ViolinPlot(data, x = "x", y = "y", add_point = TRUE)
ViolinPlot(data, x = "x", y = "y", add_trend = TRUE)
ViolinPlot(data, x = "x", y = "y", add_stat = mean)
ViolinPlot(data, x = "x", y = "y", add_bg = TRUE)
ViolinPlot(data, x = "x", y = "y", add_line = 0)
ViolinPlot(data, x = "x", y = "y", group_by = "group1")
ViolinPlot(data,
    x = "x", y = "y", group_by = "group1",
    facet_by = "group2", add_box = TRUE
)
ViolinPlot(data, x = "x", y = "y", add_point = TRUE, highlight = 'group1 == "g1"',
    alpha = 0.8, highlight_size = 1.5, pt_size = 1, add_box = TRUE)
ViolinPlot(data,
    x = "x", y = "y", group_by = "group1",
    comparisons = TRUE, sig_label = "p = {p}"
)
ViolinPlot(data,
    x = "x", y = "y", sig_label = "p.format", hide_ns = TRUE,
    facet_by = "group2", comparisons = list(c("D", "E"))
)
ViolinPlot(data,
```

```
    x = "x", y = "y", fill_mode = "mean",
    facet_by = "group2", palette = "Blues", multiplegroup_comparisons = TRUE
)
ViolinPlot(data,
    x = "x", y = "y", fill_mode = "mean",
    split_by = "group1", palette = c(g1 = "Blues", g2 = "Reds")
)
ViolinPlot(data,
    x = "x", y = "y", stack = TRUE,
    facet_by = "group2", add_box = TRUE, add_bg = TRUE,
    bg_palette = "Paired"
)


# Beeswarm plot examples
BeeswarmPlot(data, x = "x", y = "y")
BeeswarmPlot(data, x = "x", y = "y", pt_size = 1)
BeeswarmPlot(data, x = "x", y = "y", add_box = TRUE, pt_color = "grey30")
# Equivalent to:
# BoxPlot(data, x = "x", y = "y", add_beeswarm = TRUE, pt_color = "grey30")

BeeswarmPlot(data, x = "x", y = "y", group_by = "group1")
# no dodging
BeeswarmPlot(data, x = "x", y = "y", group_by = "group1", beeswarm_dodge = NULL)

BeeswarmPlot(data,
    x = "x", y = "y", beeswarm_method = "hex",
    beeswarm_cex = 2
)
```

ChordPlot *Chord / Circos plot*

Description

ChordPlot is used to create a chord plot to visualize the relationships between two categorical variables. CircosPlot is an alias of ChordPlot.

Usage

```
ChordPlot(
  data,
  y = NULL,
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  split_by = NULL,
```

```
  split_by_sep = "_",
  flip = FALSE,
  links_color = c("from", "to"),
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 0.5,
  labels_rot = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  keep_na = FALSE,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

CircosPlot(
  data,
  y = NULL,
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  links_color = c("from", "to"),
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 0.5,
  labels_rot = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  keep_na = FALSE,
```

```
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
y	A character string specifying the column name of the data frame to plot for the y-axis.
from	A character string of the column name to plot for the source. A character/factor column is expected.
from_sep	A character string to concatenate the columns in from, if multiple columns are provided.
to	A character string of the column name to plot for the target. A character/factor column is expected.
to_sep	A character string to concatenate the columns in to, if multiple columns are provided.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
flip	A logical value to flip the source and target.
links_color	A character string to specify the color of the links. Either "from" or "to".
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
alpha	A numeric value specifying the transparency of the plot.
labels_rot	A logical value to rotate the labels by 90 degrees.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

seed
: The random seed to use. Default is 8525.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

- `FALSE` (default): Drop empty factor levels from the data before plotting.
- `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
- `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

axes
: A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides
: A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.

- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A combined plot or a list of plots

Examples

```
set.seed(8525)
data <- data.frame(
    nodes1 = sample(c("Soure1", "Source2", "Source3"), 10, replace = TRUE),
    nodes2 = sample(letters[1:3], 10, replace = TRUE),
    y = sample(1:5, 10, replace = TRUE)
)

ChordPlot(data, from = "nodes1", to = "nodes2")
ChordPlot(data, from = "nodes1", to = "nodes2",
          links_color = "to", labels_rot = TRUE)
ChordPlot(data, from = "nodes1", to = "nodes2", y = "y")
ChordPlot(data, from = "nodes1", to = "nodes2", split_by = "y")
ChordPlot(data, from = "nodes1", to = "nodes2", split_by = "y",
          palette = c("1" = "Reds", "2" = "Blues", "3" = "Greens", "4" = "Purp"))
ChordPlot(data, from = "nodes1", to = "nodes2", flip = TRUE)
```

ClustreePlot *Clustree Plot*

Description

A plot visualizing Clusterings at Different Resolutions

Usage

```
ClustreePlot(
  data,
  prefix,
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
```

```
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  edge_palette = "Spectral",
  edge_palcolor = NULL,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  expand = c(0.1, 0.1),
  theme = "theme_this",
  theme_args = list(),
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
prefix	A character string of the prefix of the columns to plot. The columns with the prefix will be used to plot the tree.
flip	A logical value to flip the tree.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
edge_palette	A character string of the palette name to color the edges.
edge_palcolor	A character vector of colors to color the edges.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.

`legend.position`
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

`legend.direction`
: A character string specifying the direction of the legend.

`title`
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

`subtitle`
: A character string specifying the subtitle of the plot.

`xlab`
: A character string specifying the x-axis label.

`ylab`
: A character string specifying the y-axis label.

`expand`
: The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

`theme`
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

`theme_args`
: A list of arguments to pass to the theme function.

`combine`
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

`nrow`
: A numeric value specifying the number of rows in the facet.

`ncol`
: A numeric value specifying the number of columns in the facet.

`byrow`
: A logical value indicating whether to fill the plots by row.

`seed`
: The random seed to use. Default is 8525.

`axes`
: A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles`
: A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.

- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
N = 100
data <- data.frame(
    p.0.4 = sample(LETTERS[1:5], N, replace = TRUE),
    p.0.5 = sample(LETTERS[1:6], N, replace = TRUE),
    p.0.6 = sample(LETTERS[1:7], N, replace = TRUE),
    p.0.7 = sample(LETTERS[1:8], N, replace = TRUE),
    p.0.8 = sample(LETTERS[1:9], N, replace = TRUE),
    p.0.9 = sample(LETTERS[1:10], N, replace = TRUE),
    p.1 = sample(LETTERS[1:30], N, replace = TRUE),
    split = sample(1:2, N, replace = TRUE)
)

ClustreePlot(data, prefix = "p")
ClustreePlot(data, prefix = "p", flip = TRUE)
ClustreePlot(data, prefix = "p", split_by = "split")
ClustreePlot(data, prefix = "p", split_by = "split",
             palette = c("1" = "Set1", "2" = "Paired"))
```

`CorPairsPlot` *CorPairsPlot*

Description

Generate a grid of scatter correlation plots for all pairs of variables.

Usage

```
CorPairsPlot(
  data,
  columns = NULL,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  diag_type = NULL,
  diag_args = list(),
  layout = c(".\\", "\\.", "/.", "./"),
  cor_method = c("pearson", "spearman", "kendall"),
  cor_palette = "RdBu",
  cor_palcolor = NULL,
  cor_size = 3,
  cor_format = "corr: {round(corr, 2)}",
  cor_fg = "black",
  cor_bg = "white",
  cor_bg_r = 0.1,
  theme = "theme_this",
  theme_args = list(),
  palette = ifelse(is.null(group_by), "Spectral", "Paired"),
  palcolor = NULL,
  palreverse = FALSE,
  title = NULL,
  subtitle = NULL,
  facet_by = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
columns	The column names of the data to be plotted. If NULL, all columns, except group_by, will be used.

group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	The name of the group in the legend.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
diag_type	The type of the diagonal plots. Available types: "density", "violin", "histogram", "box", "none".
diag_args	A list of additional arguments to be passed to the diagonal plots.
layout	The layout of the plots. Available layouts: ".\", "\.", "/.", "./". • '\' or '/' means the diagonal plots are on the top-left to bottom-right diagonal. • '.' means where the scatter plots are.
cor_method	The method to calculate the correlation. Available methods: "pearson", "spearman", "kendall". The correlation will be shown in the other triangle of the scatter plots.
cor_palette	The color palette for the correlation tile plots.
cor_palcolor	Custom colors used to create a color palette for the correlation tile plots.
cor_size	The size of the correlation text.
cor_format	The format of the correlation text. Default is "corr: %.2f". It will be formatted using sprintf(cor_format, corr).
cor_fg	The color of the correlation text.
cor_bg	The background color of the correlation text.
cor_bg_r	The radius of the background of the correlation text.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

axes
: A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
  - 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides
: A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'collect' will collect guides below to the given nesting level, removing duplicates.
  - 'keep' will stop collection at this level and let guides be placed alongside their plot.
  - 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design
: Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

...
: Additional arguments.

Value

A `patch_work::wrap_plots` object or a list of them if `combine` is `FALSE`.

Examples

```
set.seed(8525)
data <- data.frame(x = rnorm(100))
data$y <- rnorm(100, 10, sd = 0.5)
data$z <- -data$x + data$y + rnorm(100, 20, 1)
data$g <- sample(1:4, 100, replace = TRUE)

CorPairsPlot(data, diag_type = "histogram", diag_args = list(bins = 30, palette = "Paired"),
 layout = "/.")

CorPairsPlot(data, group_by = "g", diag_type = "none", layout = "./",
 theme_args = list(axis.title = element_textbox(
     color = "black", box.color = "grey20", size = 16, halign = 0.5, fill = "grey90",
     linetype = 1, width = grid::unit(1, "npc"), padding = ggplot2::margin(5, 5, 5, 5))))

CorPairsPlot(data, group_by = "g", diag_type = "violin", layout = "\\.",
  cor_format = "{x}\n{y}\ncorr: {round(corr, 2)}")

CorPairsPlot(data, split_by = "g", diag_type = "none", layout = ".\\",
 legend.position = "bottom", legend.direction = "horizontal", group_name = "group")

CorPairsPlot(data, split_by = "g",
 palcolor = list("1" = "red", "2" = "blue", "3" = "green", "4" = "yellow"))
```

CorPlot *CorPlot*

Description

Generate scatter correlation plot for two variables.

Usage

```
CorPlot(
  data,
  x,
  y,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  pt_size = 2,
  pt_shape = 16,
  raster = FALSE,
  alpha = 1,
  raster_dpi = c(512, 512),
```

```
  highlight = NULL,
  highlight_color = "black",
  highlight_size = 1,
  highlight_alpha = 1,
  highlight_stroke = 0.8,
  anno_items = c("eq", "r2", "p"),
  anno_size = 3,
  anno_fg = "black",
  anno_bg = "white",
  anno_bg_r = 0.1,
 anno_position = c("topleft", "topright", "bottomleft", "bottomright", "tl", "tr", "bl",
    "br"),
  add_smooth = TRUE,
  smooth_color = "red2",
  smooth_width = 1.5,
  smooth_se = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = ifelse(is.null(group_by), "Spectral", "Paired"),
  palcolor = NULL,
  palreverse = FALSE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.

x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	The name of the group in the legend.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
pt_size	The size of the points.
pt_shape	The shape of the points.
raster	Whether to use raster graphics for plotting.
alpha	A numeric value specifying the transparency of the plot.
raster_dpi	The DPI of the raster graphics.
highlight	The items to be highlighted. Could be either a vector of rownames if data has rownames, or a vector of indices, or An expression that can be evaluated by dplyr::filter to get the highlighted items.
highlight_color	The color of the highlighted points.
highlight_size	The size of the highlighted points.
highlight_alpha	The alpha of the highlighted points.
highlight_stroke	The stroke of the highlighted points.
anno_items	The items to be annotated on the plot. Available items: "eq", "r2", "p", "spearman", "pearson", "kendall", "n".
anno_size	The size of the annotation text.
anno_fg	The color of the annotation text.
anno_bg	The background color of the annotation text.
anno_bg_r	The radius of the background of the annotation text.
anno_position	The position of the annotation text. Available positions: "topleft", "topright", "bottomleft", "bottomright". Shortcuts: "tl", "tr", "bl", "br".
add_smooth	Whether to add a linear regression line.
smooth_color	The color of the regression line.
smooth_width	The width of the regression line.
smooth_se	Whether to add the standard error band to the regression line.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

seed — The random seed to use. Default is 8525.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A ggplot object or a list of ggplot objects if combine is FALSE.

Examples

```
data(iris)
CorPlot(iris, "Sepal.Length", "Sepal.Width", group_by = "Species")
CorPlot(iris, "Sepal.Length", "Sepal.Width", group_by = "Species",
 highlight = 'Species == "setosa"', highlight_stroke = 1.5,
 anno_items = c("eq", "pearson"), anno_position = "bottomright")
CorPlot(iris, "Sepal.Length", "Sepal.Width", facet_by = "Species", facet_scales = "free")
CorPlot(iris, "Sepal.Length", "Sepal.Width", split_by = "Species",
        palette = c(setosa = "Set1", versicolor = "Dark2", virginica = "Paired"))
```

DensityPlot *Density Plot / Histogram*

Description

Density plot and histogram to illustrate the distribution of the data.

Usage

```
DensityPlot(
  data,
  x,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  xtrans = "identity",
  ytrans = "identity",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  position = "identity",
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 0.5,
  theme = "theme_this",
  theme_args = list(),
  add_bars = FALSE,
  bar_height = 0.025,
  bar_alpha = 1,
  bar_width = 0.1,
  keep_na = FALSE,
  keep_empty = FALSE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  expand = c(bottom = 0, left = 0, right = 0),
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = ifelse(is.null(group_by), "none", "right"),
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
```

```
)

Histogram(
  data,
  x,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  xtrans = "identity",
  ytrans = "identity",
  split_by = NULL,
  split_by_sep = "_",
  flip = FALSE,
  bins = NULL,
  binwidth = NULL,
  trend_skip_zero = FALSE,
  add_bars = FALSE,
  bar_height = 0.025,
  bar_alpha = 1,
  bar_width = 0.1,
  position = "identity",
  keep_na = FALSE,
  keep_empty = FALSE,
  use_trend = FALSE,
  add_trend = FALSE,
  trend_alpha = 1,
  trend_linewidth = 0.8,
  trend_pt_size = 1.5,
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 0.5,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  expand = c(bottom = 0, left = 0, right = 0),
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = ifelse(is.null(group_by), "none", "right"),
  legend.direction = "vertical",
  seed = 8525,
```

```
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of group_by
xtrans	A character string specifying the transformation of the x-axis. Default is "identity". Other options see transform of scale_x_continuous.
ytrans	A character string specifying the transformation of the y-axis. Default is "identity". Other options see transform of scale_y_continuous.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
flip	A logical value. If TRUE, the plot will be flipped.
position	How should we position the values in each bin? Default is "identity". Unlike the default position = "stack" in ggplot2::geom_histogram or ggplot2::geom_density, the default position is "identity" to show the actual count or density for each group.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
alpha	A numeric value specifying the transparency of the plot.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.

add_bars
: A logical value. If TRUE, add lines to the plot to show the data distribution on the bottom.

bar_height
: A numeric value specifying the height of the bars. The actual height will be calculated based on the maximum density or count.

bar_alpha
: A numeric value specifying the alpha of the bars.

bar_width
: A numeric value specifying the width of the bars.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

  - `FALSE` (default): Drop empty factor levels from the data before plotting.
  - `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
  - `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

expand
: The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

axes
: A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
  - 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides
: A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'collect' will collect guides below to the given nesting level, removing duplicates.
  - 'keep' will stop collection at this level and let guides be placed alongside their plot.
  - 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design
: Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

...	Additional arguments.
bins	A numeric value specifying the number of bins for the histogram.
binwidth	A numeric value specifying the width of the bins for the histogram.
trend_skip_zero	A logical value. If TRUE, skip the zero count when drawing the trend line.
use_trend	A logical value. If TRUE, use trend line instead of histogram.
add_trend	A logical value. If TRUE, add trend line to the histogram.
trend_alpha	A numeric value specifying the alpha of the trend line and points
trend_linewidth	A numeric value specifying the width of the trend line
trend_pt_size	A numeric value specifying the size of the trend points

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
data <- data.frame(
    x = c(rnorm(500, -1), rnorm(500, 1)),
    group = factor(rep(c("A", NA, "C", "D"), each = 250), levels = LETTERS[1:4]),
    facet = sample(c("F1", "F2"), 1000, replace = TRUE)
)

DensityPlot(data, x = "x")
DensityPlot(data, x = "x", group_by = "group")
DensityPlot(data, x = "x", group_by = "group",
    keep_na = TRUE, keep_empty = TRUE)
DensityPlot(data, x = "x", group_by = "group",
    keep_na = TRUE, keep_empty = 'level')
DensityPlot(data, x = "x", group_by = "group", facet_by = "facet")
DensityPlot(data, x = "x", split_by = "facet", add_bars = TRUE)
DensityPlot(data, x = "x", split_by = "facet", add_bars = TRUE,
    palette = c(F1 = "Set1", F2 = "Set2"))

set.seed(8525)
data <- data.frame(
    x = sample(setdiff(1:100, c(30:36, 50:55, 70:77)), 1000, replace = TRUE),
    group = factor(rep(c("A", "B", NA, "D"), each = 250), levels = LETTERS[1:4]),
    facet = sample(c("F1", "F2"), 1000, replace = TRUE)
)

Histogram(data, x = "x")
Histogram(data, x = "x", group_by = "group")
Histogram(data, x = "x", group_by = "group", keep_na = TRUE, keep_empty = 'level')
Histogram(data, x = "x", split_by = "facet", add_bars = TRUE)
Histogram(data, x = "x", group_by = "group", add_trend = TRUE)
Histogram(data, x = "x", group_by = "group", add_trend = TRUE, trend_skip_zero = TRUE)
```

```
Histogram(data, x = "x", group_by = "group", split_by = "facet",
 use_trend = TRUE, trend_pt_size = 3)
Histogram(data, x = "x", group_by = "group", split_by = "facet",
 palette = c(F1 = "Paired", F2 = "Spectral"))
```

DimPlot *DimPLot / FeatureDimPlot*

Description

Visualizing the dimension reduction data. `FeatureDimPlot` is used to plot the feature numeric values on the dimension reduction plot.

Usage

```
DimPlot(
  data,
  dims = 1:2,
  group_by,
  group_by_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  pt_size = NULL,
  pt_alpha = 1,
  bg_color = "grey80",
  label_insitu = FALSE,
  show_stat = !identical(theme, "theme_blank"),
  label = FALSE,
  label_size = 4,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  label_repel = FALSE,
  label_repulsion = 20,
  label_pt_size = 1,
  label_pt_color = "black",
  label_segment_color = "black",
  order = c("as-is", "reverse", "high-top", "low-top", "random"),
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  add_mark = FALSE,
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_expand = unit(3, "mm"),
  mark_alpha = 0.1,
```

```
  mark_linetype = 1,
  stat_by = NULL,
  stat_plot_type = c("pie", "ring", "bar", "line"),
  stat_plot_size = 0.1,
  stat_args = list(palette = "Set1"),
  graph = NULL,
  edge_size = c(0.05, 0.5),
  edge_alpha = 0.1,
  edge_color = "grey40",
  add_density = FALSE,
  density_color = "grey80",
  density_filled = FALSE,
  density_filled_palette = "Greys",
  density_filled_palcolor = NULL,
  lineages = NULL,
  lineages_trim = c(0.01, 0.99),
  lineages_span = 0.75,
  lineages_palette = "Dark2",
  lineages_palcolor = NULL,
  lineages_arrow = arrow(length = unit(0.1, "inches")),
  lineages_linewidth = 1,
  lineages_line_bg = "white",
  lineages_line_bg_stroke = 0.5,
  lineages_whiskers = FALSE,
  lineages_whiskers_linewidth = 0.5,
  lineages_whiskers_alpha = 0.5,
  velocity = NULL,
  velocity_plot_type = c("raw", "grid", "stream"),
  velocity_n_neighbors = NULL,
  velocity_density = 1,
  velocity_smooth = 0.5,
  velocity_scale = 1,
  velocity_min_mass = 1,
  velocity_cutoff_perc = 5,
  velocity_group_palette = "Set2",
  velocity_group_palcolor = NULL,
  arrow_angle = 20,
  arrow_color = "black",
  arrow_alpha = 1,
  streamline_l = 5,
  streamline_minl = 1,
  streamline_res = 1,
  streamline_n = 15,
  streamline_width = c(0, 0.8),
  streamline_alpha = 1,
  streamline_color = NULL,
  streamline_palette = "RdYlBu",
  streamline_palcolor = NULL,
```

```
  streamline_bg_color = "white",
  streamline_bg_stroke = 0.5,
  keep_na = FALSE,
  keep_empty = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  theme = "theme_this",
  theme_args = list(),
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  raster = NULL,
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = TRUE,
  hex_bins = 50,
  hex_binwidth = NULL,
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

FeatureDimPlot(
  data,
  dims = 1:2,
  features,
  split_by = NULL,
  split_by_sep = "_",
  lower_quantile = 0,
  upper_quantile = 0.99,
```

```
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  pt_size = NULL,
  pt_alpha = 1,
  bg_color = "grey80",
  bg_cutoff = NULL,
  label_insitu = FALSE,
  show_stat = !identical(theme, "theme_blank"),
  color_name = "",
  label = FALSE,
  label_size = 4,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  label_repel = FALSE,
  label_repulsion = 20,
  label_pt_size = 1,
  label_pt_color = "black",
  label_segment_color = "black",
  order = c("as-is", "reverse", "high-top", "low-top", "random"),
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  add_mark = FALSE,
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_expand = unit(3, "mm"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  keep_na = FALSE,
  keep_empty = FALSE,
  stat_by = NULL,
  stat_plot_type = c("pie", "ring", "bar", "line"),
  stat_plot_size = 0.1,
  stat_args = list(palette = "Set1"),
  graph = NULL,
  edge_size = c(0.05, 0.5),
  edge_alpha = 0.1,
  edge_color = "grey40",
  add_density = FALSE,
  density_color = "grey80",
  density_filled = FALSE,
  density_filled_palette = "Greys",
  density_filled_palcolor = NULL,
  lineages = NULL,
  lineages_trim = c(0.01, 0.99),
  lineages_span = 0.75,
```

```
  lineages_palette = "Dark2",
  lineages_palcolor = NULL,
  lineages_arrow = arrow(length = unit(0.1, "inches")),
  lineages_linewidth = 1,
  lineages_line_bg = "white",
  lineages_line_bg_stroke = 0.5,
  lineages_whiskers = FALSE,
  lineages_whiskers_linewidth = 0.5,
  lineages_whiskers_alpha = 0.5,
  velocity = NULL,
  velocity_plot_type = c("raw", "grid", "stream"),
  velocity_n_neighbors = NULL,
  velocity_density = 1,
  velocity_smooth = 0.5,
  velocity_scale = 1,
  velocity_min_mass = 1,
  velocity_cutoff_perc = 5,
  velocity_group_palette = "Set2",
  velocity_group_palcolor = NULL,
  arrow_angle = 20,
  arrow_color = "black",
  arrow_alpha = 1,
  streamline_l = 5,
  streamline_minl = 1,
  streamline_res = 1,
  streamline_n = 15,
  streamline_width = c(0, 0.8),
  streamline_alpha = 1,
  streamline_color = NULL,
  streamline_palette = "RdYlBu",
  streamline_palcolor = NULL,
  streamline_bg_color = "white",
  streamline_bg_stroke = 0.5,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  theme = "theme_this",
  theme_args = list(),
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  raster = NULL,
```

```
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = FALSE,
  hex_bins = 50,
  hex_binwidth = NULL,
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
dims	A character vector of the column names to plot on the x, y (and optionally z) axes or a numeric vector of the column indices. When 3 dimensions are provided, a 3D interactive plot is created using plotly. Supported in 3D: group_by, features, labels, highlight, lineages, graph/network, show_stat, order. Not supported in 3D: add_mark, stat_by, add_density, velocity, hex, facet_by, raster.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using `group_by_sep` as the separator
group_by_sep	The separator for multiple group_by columns. See `group_by`
split_by	A character vector of column names to split the data and plot separately If TRUE, we will split the data by the `features`. Each feature will be plotted separately.
split_by_sep	The separator for multiple split_by columns. See `split_by`
pt_size	A numeric value of the point size. If NULL, the point size will be calculated based on the number of data points.
pt_alpha	A numeric value of the point transparency. Default is 1.
bg_color	A character string of the background or NA points. Default is "grey80".
label_insitu	Whether to place the raw labels (group names) in the center of the points with the corresponding group. Default is FALSE, which using numbers instead of raw labels.
show_stat	Whether to show the number of points in the subtitle. Default is TRUE.
label	Whether to show the labels of groups. Default is FALSE.

label_size A numeric value of the label size. Default is 4.

label_fg A character string of the label foreground color. Default is "white".

label_bg A character string of the label background color. Default is "black".

label_bg_r A numeric value of the background ratio of the labels. Default is 0.1.

label_repel Whether to repel the labels. Default is FALSE.

label_repulsion
A numeric value of the label repulsion. Default is 20.

label_pt_size A numeric value of the label point size. Default is 1.

label_pt_color A character string of the label point color. Default is "black".

label_segment_color
A character string of the label segment color. Default is "black".

order A character string to determine the order of the points in the plot.

- "as-is": no order, the order of the points in the data will be used
- "reverse": reverse the order of the points in the data.
- "high-top": points with high values on top
- "low-top": points with low values on top
- "random": random order

For high-top and low-top, the NA values will be always plotted at the bottom. This works on features as they are numeric values. When this works on group_by, the ordering and coloring will not be changed in the legend. This is only affecting the order of drawing of the points in the plot. For high-top and low-top on group_by, the levels will be sorted based on levels of the factor. So high-top will put the points with the last levels on top, and low-top will put the points with the first levels on top. The order of points within the same level will not be changed anyway. If you need precise control over the order of group_by, set the levels of the factor before plotting. See https://github.com/pwwang/scplotter/issues/29#issuecomment-3009694130 for examples.

highlight A character vector of the row names to highlight. Default is NULL.

highlight_alpha
A numeric value of the highlight transparency. Default is 1.

highlight_size A numeric value of the highlight size. Default is 1.

highlight_color
A character string of the highlight color. Default is "black".

highlight_stroke
A numeric value of the highlight stroke. Default is 0.5.

add_mark Whether to add mark to the plot. Default is FALSE.

mark_type A character string of the mark type. Default is "hull".

mark_expand A unit value of the mark expand. Default is 3mm.

mark_alpha A numeric value of the mark transparency. Default is 0.1.

mark_linetype A numeric value of the mark line type. Default is 1.

stat_by A character string of the column name to calculate the statistics. Default is NULL.

stat_plot_type A character string of the statistic plot type. Default is "pie".

stat_plot_size A numeric value of the statistic plot size. Default is 0.1.

stat_args A list of additional arguments to the statistic plot. Default is list(palette = "Set1").

graph A character string of column names or the indexes in the data for the graph data. Default is NULL. If "@graph" is provided, the graph data will be extracted from the data attribute 'graph'.

edge_size A numeric vector of the edge size range. Default is c(0.05, 0.5).

edge_alpha A numeric value of the edge transparency. Default is 0.1.

edge_color A character string of the edge color. Default is "grey40".

add_density Whether to add density plot. Default is FALSE.

density_color A character string of the density color. Default is "grey80".

density_filled Whether to fill the density plot. Default is FALSE.

density_filled_palette
A character string of the filled density palette. Default is "Greys".

density_filled_palcolor
A character vector of the filled density palette colors. Default is NULL.

lineages A character vector of the column names for lineages. Default is NULL.

lineages_trim A numeric vector of the trim range for lineages. Default is c(0.01, 0.99).

lineages_span A numeric value of the lineages span. Default is 0.75.

lineages_palette
A character string of the lineages palette. Default is "Dark2".

lineages_palcolor
A character vector of the lineages palette colors. Default is NULL.

lineages_arrow An arrow object for the lineages. Default is arrow(length = unit(0.1, "inches")).

lineages_linewidth
A numeric value of the lineages line width. Default is 1.

lineages_line_bg
A character string of the lineages line background color. Default is "white".

lineages_line_bg_stroke
A numeric value of the lineages line background stroke. Default is 0.5.

lineages_whiskers
Whether to add whiskers to the lineages. Default is FALSE.

lineages_whiskers_linewidth
A numeric value of the lineages whiskers line width. Default is 0.5.

lineages_whiskers_alpha
A numeric value of the lineages whiskers transparency. Default is 0.5.

velocity A character (integer) vector of the column names (indexes) to pull from data for velocity. Default is NULL. It can also be a data frame or matrix of the velocity embedding itself. If NULL, the velocity will not be plotted.

velocity_plot_type
A character string of the velocity plot type. Default is "raw". One of "raw", "grid", or "stream".

velocity_n_neighbors
: A numeric value of the number of neighbors to use for velocity. Default is NULL.

velocity_density
: A numeric value of the velocity density. Default is 1.

velocity_smooth
: A numeric value of the velocity smooth. Default is 0.5.

velocity_scale
: A numeric value of the velocity scale. Default is 1.

velocity_min_mass
: A numeric value of the minimum mass for velocity. Default is 1.

velocity_cutoff_perc
: A numeric value of the velocity cutoff percentage. Default is 5.

velocity_group_palette
: A character string of the velocity group palette. Default is "Set2".

velocity_group_palcolor
: A character vector of the velocity group palette colors. Default is NULL.

arrow_angle
: An optional numeric value specifying the angle of the arrowheads in degrees for velocity arrows. Default is 20.

arrow_color
: A character string specifying the color of the velocity arrowheads. Default is "black".

arrow_alpha
: A numeric value specifying the transparency of the velocity arrows. Default is 1 (fully opaque). Only works for `plot_type = "raw"` and `plot_type = "grid"`. For `plot_type = "stream"`, use `streamline_alpha` instead.

streamline_l
: An optional numeric value specifying the length of the velocity streamlines. Default is 5.

streamline_minl
: An optional numeric value specifying the minimum length of the velocity streamlines. Default is 1.

streamline_res
: An optional numeric value specifying the resolution of the velocity streamlines. Default is 1.

streamline_n
: An optional numeric value specifying the number of velocity streamlines to draw. Default is 15.

streamline_width
: A numeric vector of length 2 specifying the width of the velocity streamlines. Default is c(0, 0.8).

streamline_alpha
: A numeric value specifying the transparency of the velocity streamlines. Default is 1 (fully opaque).

streamline_color
: A character string specifying the color of the velocity streamlines.

streamline_palette
: A character string specifying the color palette to use for the velocity streamlines. Default is "RdYlBu".

streamline_palcolor
: An optional character vector specifying the colors to use for the velocity streamlines. If NULL, the colors will be generated from the streamline_palette.

streamline_bg_color
: A character string specifying the background color of the velocity streamlines. Default is "white".

streamline_bg_stroke
: A numeric value specifying the background stroke width of the velocity streamlines. Default is 0.5.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

  - `FALSE` (default): Drop empty factor levels from the data before plotting.
  - `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
  - `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

facet_by
: A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

raster Whether to raster the plot. Default is NULL.

raster_dpi A numeric vector of the raster dpi. Default is c(512, 512).

hex Whether to use hex plot. Default is FALSE.

hex_linewidth A numeric value of the hex line width. Default is 0.5.

hex_count Whether to count the hex.

hex_bins A numeric value of the hex bins. Default is 50.

hex_binwidth A numeric value of the hex bin width. Default is NULL.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

seed The random seed to use. Default is 8525.

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow A logical value indicating whether to fill the plots by row.

axes A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

`features` A character vector of the column names to plot as features.

`lower_quantile`, `upper_quantile`, `lower_cutoff`, `upper_cutoff` Vector of minimum and maximum cutoff values or quantile values for each feature.

`bg_cutoff` A numeric value to be used a cutoff to set the feature values to NA. Default is NULL.

`color_name` A character string of the color legend name. Default is "".

Value

A ggplot object or wrap_plots object or a list of ggplot objects. When `dims` has 3 elements, a plotly object is returned instead.

A ggplot object or wrap_plots object or a list of ggplot objects. When `dims` has 3 elements, a plotly object is returned instead.

See Also

`VelocityPlot`

Examples

```
data(dim_example)

DimPlot(dim_example, group_by = "clusters")
DimPlot(dim_example, group_by = "clusters", theme = "theme_blank")
DimPlot(dim_example, group_by = "clusters", theme = ggplot2::theme_classic,
    theme_args = list(base_size = 16), palette = "seurat")
DimPlot(dim_example, group_by = "clusters", raster = TRUE, raster_dpi = 50)
DimPlot(dim_example, group_by = "clusters", highlight = 1:20,
    highlight_color = "black", highlight_stroke = 2)
DimPlot(dim_example, group_by = "clusters", highlight = TRUE, facet_by = "group",
    theme = "theme_blank")
DimPlot(dim_example, group_by = "clusters", label = TRUE,
    label_size = 5, label_bg_r = 0.2)
DimPlot(dim_example, group_by = "clusters", label = TRUE, label_fg = "red",
    label_bg = "yellow", label_size = 5)
DimPlot(dim_example, group_by = "clusters", label = TRUE, label_insitu = TRUE)
```

```
DimPlot(dim_example, group_by = "clusters", add_mark = TRUE)
DimPlot(dim_example, group_by = "clusters", add_mark = TRUE, mark_linetype = 2)
DimPlot(dim_example, group_by = "clusters", add_mark = TRUE, mark_type = "ellipse")
DimPlot(dim_example, group_by = "clusters", add_density = TRUE)
DimPlot(dim_example, group_by = "clusters", add_density = TRUE, density_filled = TRUE)
DimPlot(dim_example, group_by = "clusters", add_density = TRUE, density_filled = TRUE,
    density_filled_palette = "Blues", highlight = TRUE)
DimPlot(dim_example, group_by = "clusters", stat_by = "group")
DimPlot(dim_example, group_by = "clusters", stat_by = "group",
    stat_plot_type = "bar", stat_plot_size = 0.06)
DimPlot(dim_example, group_by = "clusters", hex = TRUE)
DimPlot(dim_example, group_by = "clusters", hex = TRUE, hex_bins = 20)
DimPlot(dim_example, group_by = "clusters", hex = TRUE, hex_count = FALSE)
DimPlot(dim_example, group_by = "clusters", graph = "@graph", edge_color = "grey80")
DimPlot(dim_example, group_by = "clusters", lineages = c("stochasticbasis_1", "stochasticbasis_2"))
DimPlot(dim_example, group_by = "clusters", lineages = c("stochasticbasis_1", "stochasticbasis_2"),
    lineages_whiskers = TRUE, lineages_whiskers_linewidth = 0.1)
DimPlot(dim_example, group_by = "clusters", lineages = c("stochasticbasis_1", "stochasticbasis_2"),
    lineages_span = 0.4)
DimPlot(dim_example, group_by = "clusters",  split_by = "group",
    palette = list(A = "Paired", B = "Set1"))
# velocity plot
DimPlot(dim_example, group_by = "clusters", velocity = c("stochasticbasis_1", "stochasticbasis_2"),
    pt_alpha = 0)
DimPlot(dim_example, group_by = "clusters", velocity = 3:4,
    velocity_plot_type = "grid", arrow_alpha = 0.6)
DimPlot(dim_example, group_by = "clusters", velocity = 3:4,
    velocity_plot_type = "stream")

# 3D plots (returns a plotly object)
DimPlot(dim_example, dims = 1:3, group_by = "clusters")
DimPlot(dim_example, dims = 1:3, group_by = "clusters", label = TRUE,
    label_insitu = TRUE)
DimPlot(dim_example, dims = c("basis_1", "basis_2", "stochasticbasis_1"),
    group_by = "clusters", graph = "@graph", edge_color = "grey80")

# keep_na and keep_empty
dim_example$clusters[dim_example$clusters == "Ductal"] <- NA

DimPlot(dim_example, group_by = "clusters", keep_na = FALSE, keep_empty = TRUE)
DimPlot(dim_example, group_by = "clusters", keep_na = TRUE, keep_empty = TRUE)
DimPlot(dim_example, group_by = "clusters", keep_na = TRUE, keep_empty = FALSE)


data(dim_example)
FeatureDimPlot(dim_example, features = "stochasticbasis_1", pt_size = 2)
FeatureDimPlot(dim_example, features = "stochasticbasis_1", pt_size = 2, bg_cutoff = 0)
FeatureDimPlot(dim_example, features = "stochasticbasis_1", raster = TRUE, raster_dpi = 30)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 pt_size = 2)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1"), pt_size = 2,
 facet_by = "group")
# Can't use facet_by for multiple features
```

```
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 pt_size = 2)
# We can use split_by
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 split_by = "group", nrow = 2)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 highlight = TRUE)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 hex = TRUE, hex_bins = 15)
FeatureDimPlot(dim_example, features = c("stochasticbasis_1", "stochasticbasis_2"),
 hex = TRUE, hex_bins = 15, split_by = "group", palette = list(A = "Reds", B = "Blues"))

# 3D plots (returns a plotly object)
FeatureDimPlot(dim_example, dims = 1:3, features = "stochasticbasis_2", pt_size = 2)
FeatureDimPlot(dim_example, dims = c("basis_1", "basis_2", "stochasticbasis_1"),
 features = "stochasticbasis_2")
```

`dim_example` *An example data for dimensionality reduction plot*

Description

This dataset is generated from the scvelo (scv.datasets.pancreas()) with the scvelo run on the dataset. Then the cell embeddings and velocity embeddings are extracted (200 downsampled), which are the first 4 columns of the data frame. The fifth column is the group identifier (clusters), and the sixth column is a fake grouping variable used to visualize stats, facetting, etc. An attribute "graph" is added to the data frame, which is a square matrix of the cell-cell distances, which is used for the graph (network) on dimensionality reduction plots.

`DotPlot` *Dot Plot / Scatter Plot / Lollipop Plot*

Description

For `DotPlot`, X-axis and Y-axis could be either numeric or factor/character. When x-axis and y-axis are both numeric, the plot works as a scatter plot. `LollipopPlot` is an alias of `DotPlot` when `lollipop` = TRUE.

Usage

```
DotPlot(
  data,
  x,
  y,
  x_sep = "_",
```

```
  y_sep = "_",
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  size_name = NULL,
  fill_name = NULL,
  fill_cutoff_name = NULL,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  bg_direction = c("vertical", "horizontal", "v", "h"),
  size_by = NULL,
  fill_by = NULL,
  fill_cutoff = NULL,
  palreverse = FALSE,
  size_min = 1,
  size_max = 10,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  seed = 8525,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
```

```
)

LollipopPlot(
  data,
  x,
  y,
  y_sep = NULL,
  flip = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  size_name = NULL,
  fill_name = NULL,
  fill_cutoff_name = NULL,
  size_by = NULL,
  fill_by = NULL,
  fill_cutoff = NULL,
  palreverse = FALSE,
  size_min = 1,
  size_max = 10,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  seed = 8525,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
```

```
  ...
)
```

Arguments

data	A data frame.
x	A character vector specifying the column to use for the x-axis. A numeric column is expected.
y	A character vector specifying the column to use for the y-axis. A factor/character column is expected.
x_sep	A character vector to concatenate multiple columns in x. Default is "_".
y_sep	A character vector to concatenate multiple columns in y. Default is "_".
flip	A logical value indicating whether to flip the x and y axes. Default is FALSE.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
size_name	A character vector specifying the name for the size legend.
fill_name	A character vector specifying the name for the fill legend.
fill_cutoff_name	A character vector specifying the name for the fill cutoff legend.
add_bg	A logical value indicating whether to add a background color to the plot. Default is FALSE.
bg_palette	A character vector specifying the palette for the background color. Default is "stripe".
bg_palcolor	A character vector specifying the color for the background color.
bg_alpha	A numeric value specifying the alpha for the background color. Default is 0.2.
bg_direction	A character vector specifying the direction for the background color. Default is "vertical". Other options are "horizontal". "h" and "v" are also accepted.
size_by	Which column to use as the size of the dots. It must be a numeric column. If not provided, the size will be the count of the instances for each 'y' in 'x'. For 'ScatterPlot', it can be a single numeric value to specify the size of the dots.
fill_by	Which column to use as the fill the dots. It must be a numeric column. If not provided, all dots will be filled with the same color at the middle of the palette.
fill_cutoff	A numeric value specifying the cutoff for the fill column. By default, the fill direction is "up". If TRUE, the fill direction is "down". When the direction is "up", the values less than the cutoff will be filled with grey. When the direction is "down", the values greater than the cutoff will be filled with grey.
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
size_min	A numeric value specifying the minimum size of the dots. Default is 1.
size_max	A numeric value specifying the maximum size of the dots. Default is 10.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha — A numeric value specifying the transparency of the plot.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

x_text_angle — A numeric value specifying the angle of the x-axis text.

seed — The random seed to use. Default is 8525.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

keep_na — A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty — One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
mtcars <- datasets::mtcars
mtcars$carb <- factor(mtcars$carb)
mtcars$gear <- factor(mtcars$gear)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, add_bg = TRUE)
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, add_bg = TRUE,
        bg_direction = "h")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, facet_by = "cyl")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, facet_by = "cyl",
        facet_scales = "free_x")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, split_by = "cyl")
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, split_by = "cyl",
        palette = list("4" = "Set1", "6" = "Paired", "8" = "Reds"))
# works as a scatter plot
DotPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18, fill_cutoff_name = "Small mpgs")
# keep_na and keep_empty
mtcars$carb[mtcars$carb == "1"] <- NA
mtcars$gear[mtcars$gear == "3"] <- NA
DotPlot(mtcars, x = "carb", y = "gear", size_by = "wt",
        fill_by = "mpg", fill_cutoff = 18,
        keep_na = TRUE, keep_empty = TRUE)

LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             fill_by = "mpg")
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             fill_by = "mpg", fill_cutoff = 18, facet_by = "cyl",
             facet_scales = "free_y")
LollipopPlot(mtcars, x = "qsec", y = "drat", size_by = "wt",
             split_by = "vs", palette = list("0" = "Reds", "1" = "Blues"))
```

`element_textbox` *Theme element that add a box to the text*

Description

Code grabbed from the `ggtext` package. See the original code at: https://github.com/wilkelab/ggtext
This is used to create a text box around the text, primarily to be used in `CorPairsPlot`.

Usage

```
element_textbox(
  family = NULL,
  face = NULL,
  size = NULL,
  colour = NULL,
  fill = NULL,
  box.colour = NULL,
  linetype = NULL,
  linewidth = NULL,
  hjust = NULL,
  vjust = NULL,
  halign = NULL,
  valign = NULL,
  lineheight = NULL,
  margin = NULL,
  padding = NULL,
  width = NULL,
  height = NULL,
  minwidth = NULL,
  maxwidth = NULL,
  minheight = NULL,
  maxheight = NULL,
  r = NULL,
  orientation = NULL,
  color = NULL,
  box.color = NULL,
  debug = FALSE,
  inherit.blank = FALSE
)

## S3 method for class 'element_textbox'
element_grob(
  element,
  label = "",
  x = NULL,
  y = NULL,
  family = NULL,
  face = NULL,
  colour = NULL,
  size = NULL,
  hjust = NULL,
  vjust = NULL,
  lineheight = NULL,
  margin = NULL,
  ...
)
```

Arguments

family	Font family
face	Font face
size	Font size (in pt)
colour, color	Text color
fill	Fill color of the enclosing box
box.colour, box.color	Line color of the enclosing box (if different from the text color)
linetype	Line type of the enclosing box (like lty in base R)
linewidth	Line width of the enclosing box (measured in mm, just like size in ggplot2::element_line()).
hjust	Horizontal justification
vjust	Vertical justification
halign	Horizontal justification
valign	Vertical justification
lineheight	Line height, in multiples of the font size
padding, margin	Padding and margins around the text box. See gridtext::textbox_grob() for details.
width, height	Unit objects specifying the width and height of the textbox, as in gridtext::textbox_grob().
minwidth, minheight, maxwidth, maxheight	Min and max values for width and height. Set to NULL to impose neither a minimum nor a maximum.
r	Unit value specifying the corner radius of the box
orientation	Orientation of the text box. See gridtext::textbox_grob() for details.
debug	Not implemented.
inherit.blank	See ggplot2::margin() for details.
element	A theme element created by element_textbox().
label	Text to display in the textbox.
x, y	Position of the textbox.
...	Other arguments passed to gridtext::textbox_grob().

Value

A ggplot2 theme element that can be used inside a ggplot2::theme() call.

EnrichMap *Enrichment Map/Network*

Description

EnrichMap is a function to plot the enrichment map. EnrichNetwork is a function to plot the enrichment network.

Usage

```
EnrichMap(
  data,
  in_form = c("auto", "clusterProfiler", "clusterprofiler", "enrichr"),
  split_by = NULL,
  split_by_sep = "_",
  top_term = 10,
  metric = "p.adjust",
  layout = "fr",
  minchar = 2,
  cluster = "fast_greedy",
  show_keyword = FALSE,
  nlabel = 4,
  character_width = 50,
  mark = "ellipse",
  label = c("term", "feature"),
  labelsize = 5,
  expand = c(0.4, 0.4),
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
```

```
  guides = NULL,
  design = NULL,
  ...
)

EnrichNetwork(
  data,
  in_form = c("auto", "clusterProfiler", "clusterprofiler", "enrichr"),
  split_by = NULL,
  split_by_sep = "_",
  top_term = 10,
  metric = "p.adjust",
  character_width = 50,
  layout = "fr",
  layoutadjust = TRUE,
  adjscale = 60,
  adjiter = 100,
  blendmode = "blend",
  labelsize = 5,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame containing the data to be plotted. It should be in the format of clusterProfiler enrichment result, which includes the columns: ID, Description, GeneRatio, BgRatio, pvalue, p.adjust, qvalue, geneID and Count.

- The ID, qvalue and Count columns are optional.
- The Description is the description of the term.
- The GeneRatio is the number of genes in the term divided by the total number of genes in the input list.
- The BgRatio is the number of genes in the term divided by the total number of genes in the background list (all terms).
- The Count column, if given, should be the same as the first number in GeneRatio.

If you have enrichment results from multiple databases, you can combine them into one data frame and add a column (e.g. Database) to indicate the database. You can plot them in a single plot using the split_by argument (e.g. split_by = "Database").

in_form — A character string specifying the input format. Either "auto", "clusterProfiler", "clusterprofiler" or "enrichr". The default is "auto", which will try to infer the input format.

split_by — The column(s) to split data by and plot separately.

split_by_sep — The separator for multiple split_by columns. See split_by

top_term — An integer specifying the number of top terms to show.

metric — A character string specifying the metric to use for the size of the nodes. It is also used to order the terms when selected the top terms. Either "pvalue" or "p.adjust". The default is "p.adjust".

layout — A character string specifying the layout of the graph. Either "circle", "tree", "grid" or other layout functions in igraph.

minchar — An integer specifying the minimum number of characters to show in the keyword.

cluster — A character string specifying the clustering method. Either "fast_greedy", "walktrap", "edge_betweenness", "infomap" or other clustering functions in igraph.

show_keyword — A logical value specifying whether to show the keyword instead of Description/Term in the plot.

nlabel — An integer specifying the number of labels to show in each cluster.

character_width — The width of the characters used to wrap the keyword.

mark — A character string specifying the mark to use for the nodes. Either "ellipse", "rect", "circle", "text" or other mark functions in ggforce.

label — A character string specifying the label to show in the legend. Either "term" or "feature". The default is "term".

labelsize — A numeric value specifying the size of the label.

expand — The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are

used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

`theme` A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

`theme_args` A list of arguments to pass to the theme function.

`palette` A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

`palcolor` A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

`palreverse` A logical value indicating whether to reverse the palette. Default is FALSE.

`alpha` A numeric value specifying the transparency of the plot.

`aspect.ratio` A numeric value specifying the aspect ratio of the plot.

`legend.position`
A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

`legend.direction`
A character string specifying the direction of the legend.

`title` A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

`subtitle` A character string specifying the subtitle of the plot.

`xlab` A character string specifying the x-axis label.

`ylab` A character string specifying the y-axis label.

`seed` The random seed to use. Default is 8525.

`combine` Whether to combine the plots into one when facet is FALSE. Default is TRUE.

`nrow` A numeric value specifying the number of rows in the facet.

`ncol` A numeric value specifying the number of columns in the facet.

`byrow` A logical value indicating whether to fill the plots by row.

`axes` A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

`layoutadjust` A logical value specifying whether to adjust the layout of the network.

`adjscale` A numeric value specifying the scale of the adjustment.

`adjiter` A numeric value specifying the number of iterations for the adjustment.

`blendmode` A character string specifying the blend mode of the colors. Either "blend", "average", "multiply" and "screen".

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data(enrich_example)
EnrichMap(enrich_example)
EnrichMap(enrich_example, label = "feature")
EnrichMap(enrich_example, show_keyword = TRUE, label = "term")
EnrichMap(enrich_example, show_keyword = TRUE, label = "feature")

data(enrich_multidb_example)
EnrichMap(enrich_multidb_example, split_by = "Database")
EnrichMap(enrich_multidb_example, split_by = "Database",
          palette = list(DB1 = "Paired", DB2 = "Set1"))


EnrichNetwork(enrich_example, top_term = 5)
```

enrich_example *An example of clusterProfiler enrichment result*

Description

An example of clusterProfiler enrichment result

Examples

```
## Not run:
if (interactive()) {
  data(geneList, package="DOSE")
  de <- names(geneList)[abs(geneList) > 1.5]
 enrich_example <- clusterProfiler::enrichPathway(gene=de, pvalueCutoff = 0.05, readable=TRUE)
  enrich_example <- as.data.frame(enrich_example)
}

## End(Not run)
```

enrich_multidb_example *An example of clusterProfiler enrichment result with multiple databases*

Description

An example of clusterProfiler enrichment result with multiple databases

Examples

```
## Not run:
if (interactive()) {
  data(enrich_example, package="plotthis")
  enrich_example$Database <- "DB1"
  enrich_example2 <- enrich_example
  enrich_example2$Database <- "DB2"
  enrich_example2$ID <- paste0(enrich_example2$ID, "_DB2")
  enrich_example2$Description <- paste0(enrich_example2$Description, " (DB2)")
  enrich_multidb_example <- rbind(enrich_example, enrich_example2)
}

## End(Not run)
```

GSEASummaryPlot *GSEA plots*

Description

- GSEASummaryPlot is used to plot a summary of the results of a GSEA analysis.
- GSEAPlot is used to plot the results of a GSEA analysis.

Usage

```
GSEASummaryPlot(
  data,
  in_form = c("auto", "dose", "fgsea"),
  gene_ranks = "@gene_ranks",
  gene_sets = "@gene_sets",
  top_term = 10,
  metric = "p.adjust",
  cutoff = 0.05,
  character_width = 50,
  line_plot_size = 0.25,
  metric_name = metric,
  nonsig_name = "Insignificant",
  linewidth = 0.2,
  line_by = c("prerank", "running_score"),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  alpha = 0.6,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  seed = 8525,
  ...
)

GSEAPlot(
  data,
  in_form = c("auto", "dose", "fgsea"),
  gene_ranks = "@gene_ranks",
  gene_sets = "@gene_sets",
```

```
  gs = NULL,
  sample_coregenes = FALSE,
  line_width = 1.5,
  line_alpha = 1,
  line_color = "#6BB82D",
  n_coregenes = 10,
  genes_label = NULL,
  label_fg = "black",
  label_bg = "white",
  label_bg_r = 0.1,
  label_size = 4,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame of GSEA results For example, from `DOSE::gseDO()`. Required columns are `ID`, `Description`, `NES`, `p.adjust`, `pvalue`. The `ID` column is used to match the gene sets.
in_form	The format of the input data • `fgsea`: The input data is from the `fgsea` package. • `dose`: The input data is from the `DOSE` package. • `auto`: Automatically detect the format of the input data. When "leadingEdge" is in the input data, it will be treated as "fgsea"; otherwise, if "core_enrichment" is in the input data, it will be treated as "dose".
gene_ranks	A numeric vector of gene ranks with genes as names The gene ranks are used to plot the gene sets. If `gene_ranks` is a character vector starting with `@`, the gene ranks will be taken from the attribute of `data`.
gene_sets	A list of gene sets, typically from a record of a GMT file The names of the list should match the `ID` column of `data`. If `gene_sets` is a character vector starting with `@`, the gene sets will be taken from the attribute of `data`. The GSEA plots will be plotted for each gene set. So, the number of plots will be the number of gene sets. If you only want to plot a subset of gene sets, you can subset the `gene_sets` before passing it to this function.

top_term An integer to select the top terms

metric The metric to use for the significance of the terms Typically the column name of p values or adjusted p values. It is also used to select the top terms.

cutoff The cutoff for the significance of the terms The terms will not be filtered with this cutoff; they are only filtered by the top_term ranked by the metric. The cutoff here is used to show the significance of the terms on the plot. For the terms that are not significant, the color will be grey.

character_width
The width of the characters in the y-axis

line_plot_size The size of the line plots

metric_name The name of the metric to show in the color bar

nonsig_name The name of the legend for the nonsignificant terms

linewidth The width of the lines in the line plots

line_by The method to calculate the line plots.

- prerank: Use the gene ranks as heights to plot the line plots.
- running_score: Use the running score to plot the line plots.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

alpha A numeric value specifying the transparency of the plot.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

seed The random seed to use. Default is 8525.

... Additional arguments.

gs — The names of the gene sets to plot If NULL, all gene sets in gene_sets will be plotted.

sample_coregenes — A logical value to sample the core genes from the core_enrichment; if FALSE, the first n_coregenes will be used

line_width — The width of the line in the running score plot

line_alpha — The alpha of the line in the running score plot

line_color — The color of the line in the running score plot

n_coregenes — The number of core genes to label

genes_label — The genes to label. If set, n_coregenes will be ignored

label_fg — The color of the label text

label_bg — The background color of the label

label_bg_r — The radius of the background color of the label

label_size — The size of the label text

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

Examples

```
data(gsea_example)
GSEASummaryPlot(gsea_example)
GSEASummaryPlot(gsea_example, line_by = "running_score")
GSEASummaryPlot(gsea_example, cutoff = 0.01)


GSEAPlot(gsea_example, gene_sets = attr(gsea_example, "gene_sets")[1])
GSEAPlot(gsea_example, gene_sets = attr(gsea_example, "gene_sets")[1:4])
```

---

gsea_example *An example of GSEA result from fgsea package*

Description

An example of GSEA result from fgsea package

Examples

```
## Not run:
if (interactive()) {
 set.seed(1234)
 data(geneList, package="DOSE")
 gsea_example <- DOSE::gseDO(geneList)
 gene_ranks <- gsea_example@geneList
 gene_sets <- gsea_example@geneSets
 gsea_example_pos <- gsea_example[gsea_example$p.adjust < 0.05 & gsea_example$NES > 0, ]
 gsea_example_neg <- gsea_example[gsea_example$p.adjust < 0.05 & gsea_example$NES < 0, ]
 gsea_example <- rbind(
     gsea_example_pos[sample(1:nrow(gsea_example_pos), 5), ],
     gsea_example_pos[sample(1:nrow(gsea_example_pos), 5), ]
 )

 attr(gsea_example, "gene_ranks") <- gene_ranks
 attr(gsea_example, "gene_sets") <- gene_sets[gsea_example$ID]
}

## End(Not run)
```

Heatmap *Heatmap*

Description

Heatmap is a popular way to visualize data in matrix format. It is widely used in biology to visualize gene expression data in microarray and RNA-seq data. The heatmap is a matrix where rows represent the samples and columns represent the features. The color of each cell represents the value of the feature in the sample. The color can be continuous or discrete. The heatmap can be split by the columns or rows to show the subgroups in the data. The heatmap can also be annotated by the columns or rows to show the additional information of the samples or features.

Usage

```
Heatmap(
  data,
  values_by = NULL,
  values_fill = NA,
  name = NULL,
  in_form = c("auto", "matrix", "wide-columns", "wide-rows", "long"),
  split_by = NULL,
  split_by_sep = "_",
  rows_by = NULL,
  rows_by_sep = "_",
  rows_split_by = NULL,
  rows_split_by_sep = "_",
  columns_by = NULL,
  columns_by_sep = "_",
  columns_split_by = NULL,
  columns_split_by_sep = "_",
  rows_data = NULL,
  columns_data = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  rows_orderby = NULL,
  columns_orderby = NULL,
  columns_name = NULL,
  columns_split_name = NULL,
  rows_name = NULL,
  rows_split_name = NULL,
  palette = "RdBu",
  palcolor = NULL,
  palreverse = FALSE,
  pie_size_name = "size",
  pie_size = NULL,
  pie_values = "length",
  pie_name = NULL,
  pie_group_by = NULL,
  pie_group_by_sep = "_",
  pie_palette = "Spectral",
  pie_palcolor = NULL,
  bars_sample = 100,
```

```
    label = identity,
    label_size = 10,
    label_color = "black",
    label_name = "label",
    mark = identity,
    mark_color = "black",
    mark_size = 1,
    mark_name = "mark",
    violin_fill = NULL,
    boxplot_fill = NULL,
    dot_size = 8,
    dot_size_name = "size",
    legend_items = NULL,
    legend_discrete = FALSE,
    legend.position = "right",
    legend.direction = "vertical",
    lower_quantile = 0,
    upper_quantile = 0.99,
    lower_cutoff = NULL,
    upper_cutoff = NULL,
    add_bg = FALSE,
    bg_alpha = 0.5,
    add_reticle = FALSE,
    reticle_color = "grey",
    cluster_columns = NULL,
    cluster_rows = NULL,
    show_row_names = NULL,
    show_column_names = NULL,
    border = TRUE,
    title = NULL,
    column_title = NULL,
    row_title = NULL,
    na_col = "grey85",
    row_names_side = "right",
    column_names_side = "bottom",
    column_annotation = NULL,
    column_annotation_side = "top",
    column_annotation_palette = "Paired",
    column_annotation_palcolor = NULL,
    column_annotation_type = "auto",
    column_annotation_params = list(),
    column_annotation_agg = NULL,
    row_annotation = NULL,
    row_annotation_side = "left",
    row_annotation_palette = "Paired",
    row_annotation_palcolor = NULL,
    row_annotation_type = "auto",
    row_annotation_params = list(),
```

```
  row_annotation_agg = NULL,
  flip = FALSE,
  alpha = 1,
  seed = 8525,
  padding = 15,
  base_size = 1,
  aspect.ratio = NULL,
  draw_opts = list(),
  layer_fun_callback = NULL,
 cell_type = c("tile", "bars", "label", "mark", "label+mark", "mark+label", "dot",
    "violin", "boxplot", "pie"),
  cell_agg = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame or matrix containing the data to be plotted. Based on the in_form, the data can have the following formats: • matrix: A matrix with rows and columns directly representing the heatmap. • long: A data frame in long format with columns for values, rows, and columns. • wide-rows: A data frame in wide format with columns for heatmap rows and values, and a single column for heatmap columns. • wide-columns: A data frame in wide format with columns for heatmap columns and values, and a single column for heatmap rows. • auto: Automatically inferred from the data format. When data is a matrix, in_form is set to "matrix". When columns_by has more than one column, in_form is set to "wide-columns". When rows_by has more than one column, in_form is set to "wide-rows". Otherwise, it is set to "long".
values_by	A character of column name in data that contains the values to be plotted. This is required when in_form is "long". For other formats, the values are pivoted into a column named by values_by.
values_fill	A value to fill in the missing values in the heatmap. When there is missing value in the data, the cluster_rows and cluster_columns will fail.
name	A character string to name the heatmap (will be used to rename values_by).
in_form	The format of the data. Can be one of "matrix", "long", "wide-rows", "wide-columns", or "auto". Defaults to "auto".

split_by
: A character of column name in data that contains the split information to split into multiple heatmaps. This is used to create a list of heatmaps, one for each level of the split. Defaults to NULL, meaning no split.

split_by_sep
: A character string to concat multiple columns in split_by.

rows_by
: A vector of column names in data that contains the row information. This is used to create the rows of the heatmap. When in_form is "long" or "wide-columns", this is requied, and multiple columns can be specified, which will be concatenated by rows_by_sep into a single column.

rows_by_sep
: A character string to concat multiple columns in rows_by.

rows_split_by
: A character of column name in data that contains the split information for rows.

rows_split_by_sep
: A character string to concat multiple columns in rows_split_by.

columns_by
: A vector of column names in data that contains the column information. This is used to create the columns of the heatmap. When in_form is "long" or "wide-rows", this is required, and multiple columns can be specified, which will be concatenated by columns_by_sep into a single column.

columns_by_sep
: A character string to concat multiple columns in columns_by.

columns_split_by
: A character of column name in data that contains the split information for columns.

columns_split_by_sep
: A character string to concat multiple columns in columns_split_by.

rows_data
: A data frame containing additional data for rows, which can be used to add annotations to the heatmap. It will be joined to the main data by rows_by and split_by if split_by exists in rows_data. This is useful for adding additional information to the rows of the heatmap.

columns_data
: A data frame containing additional data for columns, which can be used to add annotations to the heatmap. It will be joined to the main data by columns_by and split_by if split_by exists in columns_data. This is useful for adding additional information to the columns of the heatmap.

keep_na
: Whether we should keep NA groups in rows, columns and split_by variables. Default is FALSE. FALSE to remove NA groups; TRUE to keep NA groups. A vector of column names can also be provided to specify which columns to keep NA groups. Note that the record will be removed if any of the grouping columns has NA and is not specified to keep NA.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

  - FALSE (default): Drop empty factor levels from the data before plotting.
  - TRUE: Keep empty factor levels and show them as a separate category in the plot.
  - "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

rows_orderby
: A expression (in character) to specify how to order rows. It will be evaluated in the context of the data frame used for rows (after grouping by rows_split_by and rows_by). The expression should return a vector of the same length as the number of rows in the data frame. The default is NULL, which means no specific ordering. Can't be used with cluster_rows = TRUE. This is applied before renaming rows_by to rows_name.

columns_orderby
: A expression (in character) to specify how to order columns. It will be evaluated in the context of the data frame used for columns (after grouping by columns split_by and columns_by). The expression should return a vector of the same length as the number of rows in the data frame. The default is NULL, which means no specific ordering. Can't be used with cluster_columns = TRUE. This is applied before renaming columns_by to columns_name.

columns_name
: A character string to rename the column created by `columns_by`, which will be reflected in the name of the annotation or legend.

columns_split_name
: A character string to rename the column created by `columns_split_by`, which will be reflected in the name of the annotation or legend.

rows_name
: A character string to rename the column created by `rows_by`, which will be reflected in the name of the annotation or legend.

rows_split_name
: A character string to rename the column created by `rows_split_by`, which will be reflected in the name of the annotation or legend.

palette
: A character string specifying the palette of the heatmap cells.

palcolor
: A character vector of colors to override the palette of the heatmap cells.

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

pie_size_name
: A character string specifying the name of the legend for the pie size.

pie_size
: A numeric value or a function specifying the size of the pie chart. If it is a function, the function should take `count` as the argument and return the size.

pie_values
: A function or character that can be converted to a function by `match.arg()` to calculate the values for the pie chart. Default is "length". The function should take a vector of values as the argument and return a single value, for each group in `pie_group_by`.

pie_name
: A character string to rename the column created by `pie_group_by`, which will be reflected in the name of the annotation or legend.

pie_group_by
: A character of column name in `data` that contains the group information for pie charts. This is used to create pie charts in the heatmap when `cell_type` is `"pie"`.

pie_group_by_sep
: A character string to concat multiple columns in `pie_group_by`.

pie_palette
: A character string specifying the palette of the pie chart.

pie_palcolor
: A character vector of colors to override the palette of the pie chart.

bars_sample
: An integer specifying the number of samples to draw the bars.

label — A function to calculate the labels for the heatmap cells. It can take either 1, 3, or 5 arguments. The first argument is the aggregated value for a single cell. If it takes 3 arguments, the second and third arguments are the row and column indices of that cell. If it takes 5 arguments, the second and third arguments are the row and column indices, and the fourth and fifth arguments are the row and column names. The function should return one of:

- NA — no label is drawn for this cell.
- A character scalar — used as the label text; label_size and label_color are used for size and color.
- A named list with any of the following fields:
  - label: character scalar for the label text.
  - size: numeric pt size (overrides label_size).
  - color: character color string (overrides label_color).
  - legend: character string used as the legend entry for this cell's color/label combination.
  - order: integer controlling the position of this legend entry — smaller values appear first (top) in the legend. Entries without an order are appended after all explicitly ordered entries. For the indices, if you have the same dimension of data (same order of rows and columns) as the heatmap, you need to use ComplexHeatmap::pindex() to get the correct values.

label_size — A numeric value specifying the default size (pt) of the labels when cell_type = "label". Used as fallback when the label function does not return a size field.

label_color — A character string specifying the default color of the labels when cell_type = "label". Used as fallback when the label function does not return a color field. Default is "black".

label_name — A character string specifying the title of the label legend. Default is "label". The legend is shown automatically when the label function returns a list with a legend field for at least one cell — no extra configuration needed. Set legend.position = "none" to suppress all legends.

mark — A function to calculate the marks drawn on top of heatmap cells when cell_type = "mark". Same dispatch rules as label (1, 3, or 5 arguments). The function should return one of:

- NA — no mark is drawn for this cell.
- A character scalar — the mark type string; mark_color and mark_size are used for appearance.
- A named list with any of the following fields:
  - mark (or first unnamed element): character scalar, the mark type string.
  - size: numeric stroke width (lwd), overrides mark_size.
  - color: character color string, overrides mark_color.
  - legend: character string used as the legend entry key.
  - order: integer controlling legend entry position (smaller = higher).

Supported mark types:

- Primitives: `-` (h-line), `|` (v-line), `+` (cross), `/` (l-diag), `\` (r-diag), `x` (both diags), `o` (circle with gap), `()` (circle touching edge), `<>` (diamond).
- With rectangular border: `[]`, `[-]`, `[|]`, `[+]`, `[/]`, `[\]`, `[x]`, `[o]`, `[()]`, `[<>]`.
- With full circle: `(-)`, `(|)`, `(+)`, `(/)`, `(\)`, `(x)`, `(o)`, `(<>)`.
- With diamond: `<->`, `<|>`, `<+>`, `</>`, `<\>`, `<x>`, `<o>`.
- Octagon (standalone or wrapper): `{}`, `{-}`, `{|}`, `{+}`, `{/}`, `{\\}`, `{x}`, `{o}`, `{()}`, `{<>}`.
- Combinations: e.g. `[(|)]`, `[(-)]`, `[(+)]`, `[(/)]`, `[(\)]`, `[(x)]`, `[(o)]`, `[(<>)]`.

[]: R:%5C [x]: R:x [o]: R:o [()]: R:() [<>]: R:%3C%3E [(|)]: R:(%7C) [(-)]: R:(-) [(+)]: R:(+) [(/)]: R:(/) [(\)]: R:(%5C%5C) [(x)]: R:(x) [(o)]: R:(o) [(<>)]: R:(%3C%3E)

`mark_color` A character string specifying the default color of the marks when `cell_type = "mark"`. Used as fallback when the `mark` function does not return a `color` field. Default is `"black"`.

`mark_size` A numeric value specifying the default stroke width (lwd) of the marks when `cell_type = "mark"`. Used as fallback when the `mark` function does not return a `size` field. Default is `1`.

`mark_name` A character string specifying the title of the mark legend. Default is `"mark"`. The legend is shown automatically when the `mark` function returns a list with a `legend` field.

`violin_fill` A character vector of colors to override the fill color of the violin plot. If NULL, the fill color will be the same as the annotion.

`boxplot_fill` A character vector of colors to override the fill color of the boxplot. If NULL, the fill color will be the same as the annotion.

`dot_size` A numeric value specifying the size of the dot or a function to calculate the size from the values in the cell or a function to calculate the size from the values in the cell.

`dot_size_name` A character string specifying the name of the legend for the dot size. If NULL, the dot size legend will not be shown.

`legend_items` A numeric vector with names to specifiy the items in the main legend. The names will be working as the labels of the legend items.

`legend_discrete`
A logical value indicating whether the main legend is discrete.

`legend.position`
A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

`legend.direction`
A character string specifying the direction of the legend.

`lower_quantile`, `upper_quantile`, `lower_cutoff`, `upper_cutoff`
Vector of minimum and maximum cutoff values or quantile values for each feature. It's applied to aggregated values when aggregated values are used (e.g. plot_type tile, label, etc). It's applied to raw values when raw values are used (e.g. plot_type bars, etc).

add_bg A logical value indicating whether to add a background to the heatmap. Does not work with cell_type = "bars" or cell_type = "tile".

bg_alpha A numeric value between 0 and 1 specifying the transparency of the background.

add_reticle A logical value indicating whether to add a reticle to the heatmap.

reticle_color A character string specifying the color of the reticle.

cluster_columns
A logical value indicating whether to cluster the columns. If TRUE and columns_split_by is provided, the clustering will only be applied to the columns within the same split.

cluster_rows A logical value indicating whether to cluster the rows. If TRUE and rows_split_by is provided, the clustering will only be applied to the rows within the same split.

show_row_names A logical value indicating whether to show the row names. If TRUE, the legend of the row group annotation will be hidden.

show_column_names
A logical value indicating whether to show the column names. If TRUE, the legend of the column group annotation will be hidden.

border A logical value indicating whether to draw the border of the heatmap. If TRUE, the borders of the slices will be also drawn.

title The global (column) title of the heatmap

column_title A character string/vector of the column name(s) to use as the title of the column group annotation.

row_title A character string/vector of the column name(s) to use as the title of the row group annotation.

na_col A character string specifying the color for missing values. The default is "grey85".

row_names_side A character string specifying the side of the row names. The default is "right".

column_names_side
A character string specifying the side of the column names. The default is "bottom".

column_annotation
A character string/vector of the column name(s) to use as the column annotation. Or a list with the keys as the names of the annotation and the values as the column names.

column_annotation_side
A character string or named list specifying which side each column annotation is placed on. Accepts "top" (default) or "bottom".

- **String:** All column annotations go to that side (e.g. "bottom").
- **Named list:** Per-annotation side control. Keys are annotation names or aliases (.col, .col.split, etc.). Values are "top" or "bottom". Use the special .default key to set the side for unspecified annotations (e.g. list(.default = "top", my_anno = "bottom")).
- **Ordering within each side:** Name annotations (columns_by) are always placed closest to the heatmap body; split annotations (columns_split_by) are placed farthest away; user-defined annotations sit in between.

Note: Placing column annotations on "bottom" conflicts with legend.position = "bottom" — the legend may overlap the annotation names. Consider using a different legend position in that case.

column_annotation_palette
: A character string specifying the palette of the column annotation. The default is "Paired". Could be a list with the keys as the names of the annotation and the values as the palettes.

column_annotation_palcolor
: A character vector of colors to override the palette of the column annotation. Could be a list with the keys as the names of the annotation and the values as the palcolors.

column_annotation_type
: A character string specifying the type of the column annotation. The default is "auto". Other options are "simple", "pie", "ring", "bar", "violin", "boxplot", "density", "label". Could be a list with the keys as the names of the annotation and the values as the types. If the type is "auto", the type will be determined by the type and number of the column data. For split or name annotations, use aliases (e.g. .col.split, .col) to set the type.

  - "simple" — simple annotation via anno_simple() (for split/name annotations)
  - "label" — Text label annotation via anno_simple()/anno_block() (for split/name annotations)

column_annotation_params
: A list of parameters passed to the annotation function. Could be a list with the keys as the names of the annotation and the values as the parameters. For the name/split annotations, use aliases: .col/.cols/.column/.columns for columns_by, .col.split/.cols.split/.column.split/.columns.split for columns_split_by. Setting a key to FALSE disables that annotation. $<key>$show_legend controls the legend for that annotation. For "label" type annotations, use labels_gp to style the label text (e.g. labels_gp = grid::gpar(col = "white", fontsize = 12)). See anno_pie(), anno_ring(), anno_bar(), anno_violin(), anno_boxplot(), anno_density(), anno_simple(), anno_points(), anno_lines() and anno_block() for the parameters of each annotation function.

column_annotation_agg
: A function or named list of functions to aggregate values for each column annotation. If a single function, it applies to all annotations. If a named list, keys are annotation names. Defaults vary by annotation type: dplyr::first for "simple"/"points"/"lines", function(x) paste(unique(x), collapse = ", ") for "label", and no aggregation for others (e.g. "pie", "violin").

row_annotation
: A character string/vector of the column name(s) to use as the row annotation. Or a list with the keys as the names of the annotation and the values as the column names.

row_annotation_side
: A character string or named list specifying which side each row annotation is placed on. Accepts "left" (default) or "right".

  - **String:** All row annotations go to that side (e.g. "right").

- **Named list:** Per-annotation side control. Keys are annotation names or aliases (`.row`, `.rows.split`, etc.). Values are `"left"` or `"right"`. Use the special `.default` key to set the side for unspecified annotations (e.g. `list(.default = "left", .row = "right")`).
- **Ordering within each side:** Name annotations (`rows_by`) are always placed closest to the heatmap body; split annotations (`rows_split_by`) are placed farthest away; user-defined annotations sit in between.

`row_annotation_palette`
: A character string specifying the palette of the row annotation. The default is "Paired". Could be a list with the keys as the names of the annotation and the values as the palettes.

`row_annotation_palcolor`
: A character vector of colors to override the palette of the row annotation. Could be a list with the keys as the names of the annotation and the values as the palcolors.

`row_annotation_type`
: A character string specifying the type of the row annotation. The default is "auto". Other options are "simple", "pie", "ring", "bar", "violin", "boxplot", "density", "label". Could be a list with the keys as the names of the annotation and the values as the types. If the type is "auto", the type will be determined by the type and number of the row data. For split or name annotations, use aliases (e.g. `.rows.split`, `.row`) to set the type.

- `"simple"` — Simple annotation via `anno_simple()`. Only valid for row/column name and split label annotation
- `"label"` — Text label annotation via `anno_simple()`/`anno_block()` (for split/name annotations)

`row_annotation_params`
: A list of parameters passed to the annotation function. Could be a list with the keys as the names of the annotation and the values as the parameters. For the name/split annotations, use aliases: `.row`/`.rows` for `rows_by`, `.rows.split`/`.row.split` for `rows_split_by`. Setting a key to `FALSE` disables that annotation. `$<key>$show_legend` controls the legend. For `"label"` type row (name) annotations, use `label_rot` to control text rotation (default `-90` on the left side, `90` on the right side). For `"label"` type, use `labels_gp` to style the label text. Same structure as `column_annotation_params`.

`row_annotation_agg`
: A function or named list of functions to aggregate values for each row annotation. Same behavior as `column_annotation_agg`.

`flip`
: A logical value indicating whether to flip the heatmap. The idea is that, you can simply set `flip = TRUE` to flip the heatmap. You don't need to swap the arguments related to rows and columns, except those you specify via `...` that are passed to `ComplexHeatmap::Heatmap()` directly.

`alpha`
: A numeric value between 0 and 1 specifying the transparency of the heatmap cells.

`seed`
: The random seed to use. Default is 8525.

`padding`
: A numeric vector of length 4 specifying the padding of the heatmap in the order of top, right, bottom, left. Like padding in css. Note that it is different than the

padding argument in ComplexHeatmap::draw(), which is the padding in the order of bottom, left, top, right. It also support 1, 2, 3 values like css padding. When 1 element is provided, it will be used for all sides. When 2 elements are provided, the first one will be used for top and bottom, and the second one will be used for left and right. When 3 elements are provided, the first one will be used for top, the second one will be used for left and right, and the third one will be used for bottom. When 4 elements are provided, they will be used for top, right, bottom, and left respectively. If no unit is provided, the default unit will be "mm".

base_size
: A positive numeric scalar used as a scaling factor for the overall heatmap size. Default is 1 (no scaling). Values greater than 1 enlarge the heatmap; values less than 1 shrink it. Internally, all calculated cell dimensions are multiplied by this factor.

aspect.ratio
: A positive numeric scalar giving the height-to-width ratio of a single heatmap cell. When NULL (default), sensible per-cell_type defaults are used:

  - tile, label, dot: square cells (ratio = 1).
  - bars: wider-than-tall cells (ratio = 0.5) so individual bars are legible.
  - violin, boxplot, pie: square cells with a larger base size (0.5 in) so embedded sub-plots have enough room. Provide an explicit value to override these defaults (e.g. aspect.ratio = 2 for portrait cells, aspect.ratio = 0.5 for landscape cells). Note that for cell_type = "pie" the cells are always drawn square by ComplexHeatmap regardless of this setting; use it primarily to budget the figure size. Note that the aspect ratio is not guaranteed to be perfectly preserved; it will also be restricted by the size and height/width ratio of the entire plot itself.

draw_opts
: A named list of additional arguments passed to ComplexHeatmap::draw(). Arguments already managed internally (annotation_legend_list, padding, show_annotation_legend, annotation_legend_side, column_title) take precedence over any values supplied here. See https://jokergoo.github.io/ComplexHeatmap/reference/draw-HeatmapList-method.html for available options.

layer_fun_callback
: A function to add additional layers to the heatmap. The function should have the following arguments: j, i, x, y, w, h, fill, sr and sc. Please also refer to the layer_fun argument in ComplexHeatmap::Heatmap.

cell_type
: A character string specifying the type of the heatmap cells. The default is "tile" Other options are "bars", "label", "mark", "label+mark" (or equivalently "mark+label"), "dot", "violin", "boxplot" and "pie". Use "label+mark" to render both marks (drawn first, as background) and text labels (drawn on top) in each cell simultaneously, combining all label_* and mark_* parameters. Note that for pie chart, the values under columns specified by rows will not be used directly. Instead, the values will just be counted in different pie_group_by groups. NA values will not be counted.

cell_agg
: A function to aggregate the values in the cell, for the cell type "tile" and "label". The default is mean.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Other arguments passed to ComplexHeatmap::Heatmap() When row_names_max_width is passed, a unit is expected. But you can also pass a numeric values, with a default unit "inches", or a string like "5inches" to specify the number and unit directly. Unmatched arguments will be warned and ignored.

See Also

anno_simple, anno_points, anno_lines, anno_pie, anno_violin, anno_boxplot, anno_density

Examples

```
set.seed(8525)

matrix_data <- matrix(rnorm(60), nrow = 6, ncol = 10)
rownames(matrix_data) <- paste0("R", 1:6)
colnames(matrix_data) <- paste0("C", 1:10)
```

```
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(matrix_data)
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # use a different color palette
    # change the main legend title
    # show row names (legend will be hidden)
    # show column names
    # change the row name annotation name and side
    # change the column name annotation name
    Heatmap(matrix_data, palette = "viridis", values_by = "z-score",
       show_row_names = TRUE, show_column_names = TRUE,
       rows_name = "Features", row_names_side = "left",
       columns_name = "Samples")
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # flip the heatmap
    Heatmap(matrix_data, palette = "viridis", values_by = "z-score",
       show_row_names = TRUE, show_column_names = TRUE,
       rows_name = "Features", row_names_side = "left",
       columns_name = "Samples", flip = TRUE)
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # add annotations to the heatmap
    rows_data <- data.frame(
       rows = paste0("R", 1:6),
       group = sample(c("X", "Y", "Z"), 6, replace = TRUE)
    )
    Heatmap(matrix_data, rows_data = rows_data,
        row_annotation = list(Group = "group"),
        row_annotation_type = list(Group = "simple"),
        row_annotation_palette = list(Group = "Spectral")
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group"
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # use label annotation for split groups (shows group labels inside colored blocks)
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group",
        row_annotation_params = list(.rows.split = list(
            border = FALSE,
            labels_gp = grid::gpar(col = "white", fontsize = 12),
            labels_rot = 0
        )),
        row_annotation_type = list(.rows.split = "label")
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # label annotation for column splits
```

```
    columns_data <- data.frame(
        columns = paste0("C", 1:10),
        batch = rep(c("A", "B"), each = 5)
    )
    Heatmap(matrix_data, columns_data = columns_data,
        columns_split_by = "batch",
        column_annotation_type = list(.col.split = "label")
    )
}
rownames(matrix_data)[1] <- "R12345"
if (requireNamespace("cluster", quietly = TRUE)) {
    # label annotation for name annotations: show row/column names as colored labels
    Heatmap(matrix_data, rows_data = rows_data,
        row_annotation_type = list(.row = "label"),
        column_annotation_type = list(.col = "label"),
        column_annotation_params = list(.col = list(labels_rot = 90)),
        row_annotation_palette = list(.row = "Set2"),
        row_annotation_side = list(.row = "right"),
        row_annotation_params = list(.row = list(labels_rot = 150))
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # add labels to the heatmap
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group", cell_type = "label",
        base_size = 0.8,
        label = function(x) ifelse(
            x > 0, scales::number(x, accuracy = 0.01), NA
        )
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # add labels based on an external data
    pvalues <- matrix(runif(60, 0, 0.5), nrow = 6, ncol = 10)
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group", cell_type = "label",
        base_size = 0.8,
        label = function(x, i, j) {
            pv <- ComplexHeatmap::pindex(pvalues, i, j)
            ifelse(pv < 0.01, "***",
            ifelse(pv < 0.05, "**",
            ifelse(pv < 0.1, "*", NA)))
        }
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # Set label color, size, legend and order
    pvalues <- matrix(runif(60, 0, 0.5), nrow = 6, ncol = 10)
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group", cell_type = "label",
        base_size = 0.6,
        label_name = "Significance",
        label = function(x, i, j) {
```

```
            pv <- ComplexHeatmap::pindex(pvalues, i, j)
            if (pv < 0.01)
               list("***", color = "red", size = 12, legend = "p < 0.01", order = 1)
            else if (pv < 0.05)
               list("**", color = "orange", size = 10, legend = "p < 0.05", order = 3)
            else if (pv < 0.1)
               list("*", color = "yellow", size = 8, legend = "p < 0.1", order = 2)
            else NA
        }
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # add marks
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group", cell_type = "mark",
        mark = function(x, i, j) {
            pv <- ComplexHeatmap::pindex(pvalues, i, j)
            if(pv < 0.01) list("[x]", legend = "p < 0.01")
            else if (pv < 0.02) list("[o]", legend = "p < 0.02")
            else if (pv < 0.03) list("[-]", legend = "p < 0.03")
            else if (pv < 0.05) list("[()]", legend = "p < 0.05")
            else if (pv < 0.06) list("+", legend = "p < 0.06")
            else if (pv < 0.07) list("x", legend = "p < 0.07")
            else if (pv < 0.08) list("[/]", legend = "p < 0.08")
            else if (pv < 0.09) list("[\\]", legend = "p < 0.09")
            else NA
        }
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # add labels and marks
    Heatmap(matrix_data, rows_data = rows_data,
        rows_split_by = "group", cell_type = "mark+label",
        label = scales::label_number(accuracy = 0.01),
        mark = function(x, i, j) {
            pv <- ComplexHeatmap::pindex(pvalues, i, j)
            if(pv < 0.01) list("{}", legend = "p < 0.01")
            else if(pv < 0.05) list("[]", legend = "p < 0.05")
            else NA
        },
        mark_size = 1.5, mark_color = "red"
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # quickly simulate a GO board
    go <- matrix(sample(c(0, 1, NA), 81, replace = TRUE), ncol = 9)

    Heatmap(
        go,
        # Do not cluster rows and columns and hide the name annotations
       # Use .row/.col aliases (or the actual rows_name/columns_name) in annotation_params
        cluster_rows = FALSE, cluster_columns = FALSE,
        row_annotation_params = list(.row = FALSE),
```

```
        column_annotation_params = list(.col = FALSE),
        show_row_names = FALSE, show_column_names = FALSE,
        # Set the legend items
        values_by = "Players", legend_discrete = TRUE,
        legend_items = c("Player 1" = 0, "Player 2" = 1),
        # Set the pawns
        cell_type = "dot", dot_size = function(x) ifelse(is.na(x), 0, 10),
        dot_size_name = NULL,  # hide the dot size legend
        palcolor = c("white", "black"),
        # Set the board
        add_reticle = TRUE,
        # Set the size of the board
        width = ggplot2::unit(105, "mm"), height = ggplot2::unit(105, "mm"))
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # Make the row/column name annotation thicker using the .row/.col aliases
    Heatmap(matrix_data,
        column_annotation_params = list(.col = list(height = 5)),
        row_annotation_params = list(.row = list(width = 5)))
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # Per-annotation side control: row name annotation on the right,
    # all other row annotations on the left (.default)
    rows_data2 <- data.frame(
        rows = paste0("R", 1:6),
        group = sample(c("X", "Y"), 6, replace = TRUE),
        score = runif(6)
    )
    Heatmap(matrix_data, rows_data = rows_data2,
        rows_split_by = "group",
        row_annotation = list(Score = "score"),
        row_annotation_side = list(.default = "left", .row = "right"),
        show_row_names = TRUE
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # Move all row annotations to the right side
    Heatmap(matrix_data, rows_data = rows_data2,
        rows_split_by = "group",
        row_annotation = list(Score = "score"),
        row_annotation_side = "right",
        show_row_names = TRUE
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # Split and name annotations on opposite sides:
    # split annotation on the default left, name annotation on the right
    Heatmap(matrix_data, rows_data = rows_data2,
        rows_split_by = "group",
        row_annotation_side = list(.default = "left", .row = "right"),
        show_row_names = TRUE
    )
}
```

```
if (requireNamespace("cluster", quietly = TRUE)) {
    # Row name label annotation on the right side (text rotated 90° clockwise)
    Heatmap(matrix_data, rows_data = rows_data2,
        row_annotation_type = list(.row = "label"),
        row_annotation_palette = list(.row = "Set2"),
        row_annotation_side = list(.row = "right"),
        show_row_names = TRUE
    )
}

# Use long form data
N <- 500
data <- data.frame(
    value = rnorm(N),
    c = sample(letters[1:8], N, replace = TRUE),
    r = sample(LETTERS[1:5], N, replace = TRUE),
    p = sample(c("x", "y"), N, replace = TRUE),
    q = sample(c("X", "Y", "Z"), N, replace = TRUE),
    a = as.character(sample(1:5, N, replace = TRUE)),
    p1 = runif(N),
    p2 = runif(N)
)

if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(data, rows_by = "r", columns_by = "c", values_by = "value",
        rows_split_by = "p", columns_split_by = "q", show_column_names = TRUE)
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # split into multiple heatmaps
    Heatmap(data,
        values_by = "value", columns_by = "c", rows_by = "r", split_by = "p",
        upper_cutoff = 2, lower_cutoff = -2, legend.position = c("none", "right"),
        design = "AAAAAA#BBBBBBB"
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # cell_type = "bars" (default is "tile")
    Heatmap(data, values_by = "value", rows_by = "r", columns_by = "c",
        cell_type = "bars")
}
if (requireNamespace("cluster", quietly = TRUE)) {
    p <- Heatmap(data, values_by = "value", rows_by = "r", columns_by = "c",
        cell_type = "dot", dot_size = length, dot_size_name = "data points",
        add_bg = TRUE, add_reticle = TRUE)
    p
}
if (requireNamespace("cluster", quietly = TRUE)) {
    dot_size_data <- p@data
    # Make it big so we can see if we get the right indexing
    # for dot_size function
    dot_size_data["A", "a"] <- max(dot_size_data) * 2

    Heatmap(data, values_by = "value", rows_by = "r", columns_by = "c",
```

```
        cell_type = "dot", dot_size_name = "data points",
        dot_size = function(x, i, j) ComplexHeatmap::pindex(dot_size_data, i, j),
        show_row_names = TRUE, show_column_names = TRUE,
        add_bg = TRUE, add_reticle = TRUE)
}
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(data, values_by = "value", rows_by = "r", columns_by = "c",
        cell_type = "pie", pie_group_by = "q", pie_size = sqrt,
        add_bg = TRUE, add_reticle = TRUE)
}
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(data, values_by = "value", rows_by = "r", columns_by = "c",
        cell_type = "violin", add_bg = TRUE, add_reticle = TRUE)
}
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(data, values_by = "value", rows_by = "r", columns_by = "c",
        cell_type = "boxplot", add_bg = TRUE, add_reticle = TRUE)
}
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(data,
        values_by = "value", rows_by = "r", columns_by = "c",
        column_annotation = list(r1 = "p", r2 = "q", r3 = "p1"),
        column_annotation_type = list(r1 = "ring", r2 = "bar", r3 = "violin"),
        column_annotation_params = list(
            r1 = list(height = grid::unit(10, "mm"), show_legend = FALSE),
            r3 = list(height = grid::unit(18, "mm"))
        ),
        row_annotation = c("q", "p2", "a"),
        row_annotation_side = "right",
        row_annotation_type = list(q = "pie", p2 = "density", a = "simple"),
        row_annotation_params = list(q = list(width = grid::unit(12, "mm"))),
        show_row_names = TRUE, show_column_names = TRUE
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    Heatmap(data,
        values_by = "value", rows_by = "r", columns_by = "c",
        split_by = "p", palette = list(x = "Reds", y = "Blues")
    )
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # implies in_form = "wide-rows"
    Heatmap(data, rows_by = c("p1", "p2"), columns_by = "c")
}
if (requireNamespace("cluster", quietly = TRUE)) {
    # implies wide-columns
    Heatmap(data, rows_by = "r", columns_by = c("p1", "p2"))
}
```

JitterPlot *Jitter Plot*

Description

Jittered point plot with optional background, highlight, labels and faceting.

Usage

```
JitterPlot(
  data,
  x,
  x_sep = "_",
  y = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  keep_na = FALSE,
  keep_empty = FALSE,
  sort_x = c("none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc",
    "median"),
  flip = FALSE,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  x_text_angle = 0,
  order_by = "-({y}^2 + {size_by}^2)",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  aspect.ratio = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  shape = 21,
  border = "black",
  size_by = 2,
  size_name = NULL,
  size_trans = NULL,
  y_nbreaks = 4,
  jitter_width = 0.5,
  jitter_height = 0,
  y_max = NULL,
  y_min = NULL,
  y_trans = "identity",
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_hline = NULL,
```

```
  hline_type = "solid",
  hline_width = 0.5,
  hline_color = "black",
  hline_alpha = 1,
  labels = NULL,
  label_by = NULL,
  nlabel = 5,
  label_size = 3,
  label_fg = "black",
  label_bg = "white",
  label_bg_r = 0.1,
  highlight = NULL,
  highlight_color = "red2",
  highlight_size = 1,
  highlight_alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string of the column name to plot on the x-axis. A character/factor column is expected. If multiple columns are provided, the columns will be concatenated with x_sep.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided. When in_form is "wide", x columns will not be concatenated.
y	A character string of the column name to plot on the y-axis. A numeric column is expected. When in_form is "wide", y is not required. The values under x columns will be used as y-values.
in_form	A character string to specify the input data type. Either "long" or "wide".

split_by — The column(s) to split data by and plot separately.

split_by_sep — The separator for multiple split_by columns. See split_by

keep_na — A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty — One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

sort_x — A character string to specify the sorting of x-axis, chosen from "none", "mean_asc", "mean_desc", "mean", "median_asc", "median_desc", "median".

- none means no sorting (as-is).
- mean_asc sorts the x-axis by ascending mean of y-values.
- mean_desc sorts the x-axis by descending mean of y-values.
- mean is an alias for mean_asc.
- median_asc sorts the x-axis by ascending median of y-values.
- median_desc sorts the x-axis by descending median of y-values.
- median is an alias for median_asc.

flip — A logical value to flip the plot.

group_by — A character string to dodge the points.

group_by_sep — A character string to concatenate the columns in group_by, if multiple columns are provided.

group_name — A character string to name the legend of dodge.

x_text_angle — A numeric value specifying the angle of the x-axis text.

order_by — A string of expression passed to dplyr::arrange() to order the data to get the top nlabel points for labeling. Default is -({y}^2 + {size_by}^2) (similar to VolcanoPlot).

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

alpha Point transparency.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

shape A numeric value to specify the point shape. Shapes 21–25 have borders; border behavior is controlled by border.

border A logical or character value to specify the border of points when the shape has border (21–25). If TRUE, border color follows the point color (same as fill). If a color string, uses that constant border color. If FALSE, no border.

size_by A numeric column name or a single numeric value for the point size. When a column, sizes are scaled (see scatter plots).

size_name Legend title for size when size_by is a column.

size_trans A function or a name of a global function to transform size_by (when size_by is a column). The legend shows original (untransformed) values.

jitter_width, jitter_height
Jitter parameters.

y_max, y_min Numeric or quantile strings ("q95", "q5") for y limits computation (used for fixed coord).

y_trans, y_nbreaks
Axis settings.

add_bg A logical value to add background to the plot.

bg_palette A character string to specify the palette of the background.

bg_palcolor A character vector to specify the colors of the background.

bg_alpha A numeric value to specify the transparency of the background.

add_hline Add one or more horizontal reference lines at the given y-value(s).

hline_type The line type for the horizontal reference line(s).

hline_width The line width for the horizontal reference line(s).

hline_color The color for the horizontal reference line(s).

hline_alpha The alpha for the horizontal reference line(s).

labels A vector of row names or indices to label the points.

label_by A character column name to use as the label text. If NULL, rownames are used.

nlabel — Number of points to label per x-group when labels is NULL (top by y^2 + size^2).

label_size, label_fg, label_bg, label_bg_r — Label aesthetics.

highlight, highlight_color, highlight_size, highlight_alpha — Highlighted point options.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

seed — The random seed to use. Default is 8525.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

The Jitter plot(s). When split_by is not provided, it returns a ggplot object. When split_by is provided, it returns a object of plots wrapped by patchwork::wrap_plots if combine = TRUE; otherwise, it returns a list of ggplot objects.

Examples

```
set.seed(8525)
n <- 180
x <- factor(
    sample(c("A", NA, LETTERS[3:5]), n, replace = TRUE),
    levels = c("A", "B", "C", "D", "E")
)
group <- factor(
    sample(c("G1", NA, "G3"), n, replace = TRUE),
    levels = c("G1", "G2", "G3")
)
size <- rexp(n, rate = 1)
id <- paste0("pt", seq_len(n))
y <- rnorm(n, mean = ifelse(is.na(group), 0, ifelse(group == "G1", 0.5, -0.5))) +
     as.numeric(ifelse(is.na(x), 0, x))/10
df <- data.frame(
  x = x,
  y = y,
  group = group,
  size = size,
  id = id
)

# Basic
JitterPlot(df, x = "x", y = "y")

# Keep empty x levels and NA
JitterPlot(df, x = "x", y = "y", keep_na = TRUE, keep_empty = TRUE)
```

```
# Map size with transform; legend shows original values
JitterPlot(df, x = "x", y = "y", size_by = "size", size_name = "Abundance",
    size_trans = sqrt, order_by = "-y^2")

# Dodge by group and add a horizontal line
JitterPlot(df, x = "x", y = "y", group_by = "group",
  add_hline = 0, hline_type = "dashed", hline_color = "red2")

# Keep the empty levels only for color coding
# Note the G3 is not blue (which is taken by unused level G2)
JitterPlot(df, x = "x", y = "y", group_by = "group",
    keep_na = TRUE, keep_empty = 'level')

# Label top points by distance (y^2 + size^2)
JitterPlot(df, x = "x", y = "y", size_by = "size", label_by = "id", nlabel = 3)

# Flip axes
JitterPlot(df, x = "x", y = "y", flip = TRUE)
```

LinePlot *Line Plot*

Description

Visualizing the change of a numeric value over the progression of a categorical variable.

Usage

```
LinePlot(
  data,
  x,
  y = NULL,
  group_by = NULL,
  group_by_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  fill_point_by_x_if_no_group = TRUE,
  color_line_by_x_if_no_group = TRUE,
  add_bg = FALSE,
  bg_palette = "stripe",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  add_errorbars = FALSE,
  errorbar_width = 0.1,
  errorbar_alpha = 1,
  errorbar_color = "grey30",
  errorbar_linewidth = 0.75,
  errorbar_min = NULL,
```

```
  errorbar_max = NULL,
  errorbar_sd = NULL,
  highlight = NULL,
  highlight_size = pt_size - 0.75,
  highlight_color = "red2",
  highlight_alpha = 0.8,
  pt_alpha = 1,
  pt_size = 5,
  keep_na = FALSE,
  keep_empty = FALSE,
  line_type = "solid",
  line_width = 1,
  line_alpha = 0.8,
  add_hline = FALSE,
  hline_type = "solid",
  hline_width = 0.5,
  hline_color = "black",
  hline_alpha = 1,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  facet_by = NULL,
  facet_scales = "fixed",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  facet_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	A character string specifying the separator to use when concatenating multiple columns.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
fill_point_by_x_if_no_group	A logical value indicating whether to color the points by the x-axis values when there is no group_by column.
color_line_by_x_if_no_group	A logical value indicating whether to color the lines by the x-axis values
add_bg	A logical value indicating whether to add a background to the plot.
bg_palette	The palette to use for the background.
bg_palcolor	The color to use for the background.
bg_alpha	The alpha value of the background.
add_errorbars	A logical value indicating whether to add error bars to the plot.
errorbar_width	The width of the error bars.
errorbar_alpha	The alpha value of the error bars.
errorbar_color	The color to use for the error bars. If "line", the error bars will be colored the same as the lines.
errorbar_linewidth	The line width of the error bars.
errorbar_min	The column in the data frame containing the lower bound of the error bars.
errorbar_max	The column in the data frame containing the upper bound of the error bars.
errorbar_sd	The column in the data frame containing the standard deviation of the error bars. If errorbar_min and errorbar_max are not provided, this column will be used to calculate the error bars. errorbar_min = y - errorbar_sd, errorbar_max = y + errorbar_sd. If errorbar_min and errorbar_max are provided, this column will be ignored.
highlight	A vector of indexes or rownames to select the points to highlight. It could also be an expression (in string) to filter the data.
highlight_size	The size of the highlighted points.
highlight_color	A character vector specifying the color of the highlighted points. Default is "red".

highlight_alpha
: A numeric value specifying the transparency of the highlighted points. Default is 1.

pt_alpha
: The alpha value of the points.

pt_size
: The size of the points.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

  - `FALSE` (default): Drop empty factor levels from the data before plotting.
  - `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
  - `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

line_type
: The type of line to draw.

line_width
: The width of the line.

line_alpha
: The alpha value of the line.

add_hline
: A numeric value indicating the y-intercept of a horizontal line to add to the plot. If FALSE, no horizontal line will be added.

hline_type
: The type of line to draw for the horizontal line.

hline_width
: The width of the horizontal line.

hline_color
: The color of the horizontal line. When `group_by` is provided, this can be TRUE to use the same color as the lines.

hline_alpha
: The alpha value of the horizontal line.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

x_text_angle A numeric value specifying the angle of the x-axis text.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow A logical value indicating whether to fill the plots by row.

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

facet_args A list of arguments to pass to `ggplot2::facet_wrap()` or `ggplot2::facet_grid()`. when there is no group_by column.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

seed The random seed to use. Default is 8525.

axes A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.

- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
   x = factor(c("A", "B", "C", "D", "A", "B", "C", "D"), levels = LETTERS[1:6]),
   y = c(10, 8, 16, 4, 6, 12, 14, 2),
   group = c("G1", "G1", "G1", "G1", "G2", "G2", "G2", "G2"),
   facet = c("F1", "F1", "F2", "F2", "F3", "F3", "F4", "F4")
)

LinePlot(data, x = "x", y = "y")
LinePlot(data, x = "x", y = "y", highlight = "group == 'G1'",
   fill_point_by_x_if_no_group = FALSE, color_line_by_x_if_no_group = FALSE)
LinePlot(data, x = "x", y = "y", group_by = "group")
LinePlot(data, x = "x", y = "y", group_by = "group",
   add_hline = 10, hline_color = "red")
LinePlot(data, x = "x", y = "y", group_by = "group", add_bg = TRUE,
   highlight = "y > 10")
LinePlot(data, x = "x", y = "y", group_by = "group", facet_by = "facet")
LinePlot(data, x = "x", y = "y", group_by = "group", split_by = "facet")
LinePlot(data, x = "x", y = "y", split_by = "group",
         palcolor = list(G1 = c("red", "blue"), G2 = c("green", "black")))

# keep_na and keep_empty
data <- data.frame(
   x = factor(c("A", "B", NA, "D", "A", "B", NA, "D"), levels = LETTERS[1:4]),
   y = c(10, 8, 16, 4, 6, 12, 14, 2),
   group = factor(c("G1", "G1", "G1", NA, NA, "G3", "G3", "G3"),
```

```
      levels = c("G1", "G2", "G3")),
   facet = c("F1", "F1", "F2", "F2", "F3", "F3", "F4", "F4")
)

LinePlot(data, x = "x", y = "y", keep_na = TRUE)
LinePlot(data, x = "x", y = "y", keep_empty = TRUE)
LinePlot(data, x = "x", y = "y", keep_empty = 'level')
LinePlot(data, x = "x", y = "y", group_by = "group", keep_na = TRUE)
LinePlot(data, x = "x", y = "y", group_by = "group", keep_empty = TRUE)
LinePlot(data, x = "x", y = "y", group_by = "group",
   keep_empty = list(x = TRUE, group = 'level'))
```

ManhattanPlot *ManhattanPlot*

Description

This function is borrowed from ggmanh::manhattan_plot() with following customizations:

Usage

```
ManhattanPlot(
  data,
  chr_by,
  pos_by,
  pval_by,
  split_by = NULL,
  split_by_sep = "_",
  label_by = NULL,
  chromosomes = NULL,
  pt_size = 0.75,
  pt_color = NULL,
  pt_alpha = alpha,
  pt_shape = 19,
  label_size = 3,
  label_fg = NULL,
  highlight = NULL,
  highlight_color = NULL,
  highlight_size = 1.5,
  highlight_alpha = 1,
  highlight_shape = 19,
  preserve_position = TRUE,
  chr_gap_scaling = 1,
  pval_transform = "-log10",
  signif = c(5e-08, 1e-05),
  signif_color = NULL,
  signif_rel_pos = 0.2,
```

```
  signif_label = TRUE,
  signif_label_size = 3.5,
  signif_label_pos = c("left", "right"),
  thin = NULL,
  thin_n = 1000,
  thin_bins = 200,
  rescale = TRUE,
  rescale_ratio_threshold = 5,
  palette = "Dark2",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = expression("-" * log[10](p)),
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  facet_by = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
chr_by	Column name for chromosome (default: "chr").
pos_by	Column name for position (default: "pos").
pval_by	Column name for p-value (default: "pval").
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See `split_by`
label_by	Column name for the variants to be labeled (default: NULL). Only the variants with values in this column will be labeled.
chromosomes	A vector of chromosomes to be plotted (default: NULL). If NULL, all chromosomes will be plotted. It is more of a combination of the `chromosome` and `chr.order` arguments of `ggmanh::manhattan_plot()`. We can use it to select chromosomes to be plotted or to set the order of the chromosomes.
pt_size	A numeric value to specify the size of the points in the plot.

pt_color A character string to specify the color of the points in the plot. By default, the color of the points will be controled by palette or palcolor arguments. This is useful to color the background points when highlight and highlight_color are specified.

pt_alpha A numeric value to specify the transparency of the points in the plot.

pt_shape A numeric value to specify the shape of the points in the plot.

label_size A numeric value to specify the size of the labels in the plot.

label_fg A character string to specify the color of the labels in the plot. If NULL, the color of the labels will be the same as the points.

highlight Either a vector of indices or a character of expression to select the variants to be highlighted (default: NULL). If NULL, no variants will be highlighted.

highlight_color
A character string to specify the color of the highlighted points.

highlight_size A numeric value to specify the size of the highlighted points.

highlight_alpha
A numeric value to specify the transparency of the highlighted points.

highlight_shape
A numeric value to specify the shape of the highlighted points.

preserve_position
If TRUE, the width of each chromosome reflect the number of variants and the position of each variant is correctly scaled? If FALSE, the width of each chromosome is equal and the variants are equally spaced.

chr_gap_scaling
A numeric value to specify the scaling of the gap between chromosomes. It is used to adjust the gap between chromosomes in the plot.

pval_transform A function to transform the p-values (default: -log10). If it is a character, it will be evaluated as a function.

signif A vector of significance thresholds (default: c(5e-08, 1e-05)).

signif_color A character vector of equal length as signif. It contains colors for the lines drawn at signif. If NULL, the smallest value is colored black while others are grey.

signif_rel_pos A numeric between 0.1 and 0.9. If the plot is rescaled,

signif_label A logical value indicating whether to label the significance thresholds (default: TRUE).

signif_label_size
A numeric value to specify the size of the significance labels.

signif_label_pos
A character string specifying the position of the significance labels. where should the significance threshold be positioned? It can be either "left" or "right" (default: "left").

thin A logical value indicating whether to thin the data (default: NULL). Defaults to TRUE when chromosomes is specified and the length of it is less than the number of chromosomes in the data. Defaults to FALSE otherwise.

thin_n Number of max points per horizontal partitions of the plot. Defaults to 1000.

thin_bins — Number of bins to partition the data. Defaults to 200.

rescale — A logical value indicating whether to rescale the plot (default: TRUE).

rescale_ratio_threshold — A numeric value to specify the ratio threshold for rescaling.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

alpha — A numeric value specifying the transparency of the plot.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

seed — The random seed to use. Default is 8525.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`facet_by` A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Details

- The dots in argument names are replaced with underscores wherever possible.
- `chr.colname`, `pos.colname`, `pval.colname` and `label.colname` are replaced with `chr_by`, `pos_by`, `pval_by` and `label_by` respectively.
- The `chromosome` and `chr.order` arguments are merged into a single argument `chromosomes`.
- The `highlight.colname` argument is replaced with `highlight`, which can be a vector of indices or a character of expression to select the variants to be highlighted, instead of a column name.
- `point.size` is replaced with `pt_size`
- When `highlight` is specified, the colors of the points will be controled by `pt_color` and `highlight_color` arguments.
- The labels get more controled by `label_*` arguments.
- The highlighted points get more controled by `highlight_*` arguments.
- The `pval_log_transform` argument is replaced with `pval_transform`, which allows to specify a function to transform the p-values.

Value

A ggplot object or wrap_plots object or a list of ggplot objects. If no `split_by` is provided, a single plot (ggplot object) will be returned. If 'combine' is TRUE, a wrap_plots object will be returned. If 'combine' is FALSE, a list of ggplot objects will be returned.

Examples

```
set.seed(1000)

nsim <- 50000

simdata <- data.frame(
  "chromosome" = sample(c(1:22,"X"), size = nsim, replace = TRUE),
```

```
  "position" = sample(1:100000000, size = nsim),
  "P.value" = rbeta(nsim, shape1 = 5, shape2 = 1)^7,
  "cohort" = sample(c("A", "B"), size = nsim, replace = TRUE)
)
simdata$chromosome <- factor(simdata$chromosome, c(1:22, "X"))
options(repr.plot.width=10, repr.plot.height=5)

if (requireNamespace("ggmanh", quietly = TRUE)) {
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   title = "Simulated P.Values", ylab = "P")
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# split_by
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   title = "Simulated P.Values", ylab = "P", split_by = "cohort", ncol = 1)
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Customized p-value transformation and significance threshold line colors
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   title = "Simulated -Log2 P.Values", ylab = "-log2(P)", pval_transform = "-log2",
   signif_color = c("red", "blue"))
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Use a different palette and don't show significance threshold labels
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   palette = "Set1", signif_label = FALSE)
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Reverse the palette and show significance threshold labels on the right
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   palette = "Set1", palreverse = TRUE, signif_label_pos = "right")
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Use chromosomes to show a single selected chromosome
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
   title = "Simulated P.Values", chromosomes = 5)
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Subset and reorder chromosomes
ManhattanPlot(
   simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
```

```
   title = "Simulated P.Values", chromosomes = c(20, 4, 6))
}

tmpdata <- data.frame(
  "chromosome" = c(rep(5, 10), rep(21, 5)),
  "position" = c(sample(250000:250100, 10, replace = FALSE),
    sample(590000:600000, 5, replace = FALSE)),
  "P.value" = c(10^-(rnorm(10, 100, 3)), 10^-rnorm(5, 9, 1)),
  "cohort" = c(rep("A", 10), rep("B", 5))
)

simdata <- rbind(simdata, tmpdata)
simdata$chromosome <- factor(simdata$chromosome, c(1:22, "X"))

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Don't rescale the plot (y-axis)
ManhattanPlot(
    simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    title = "Simulated P.Values - Significant", rescale = FALSE)
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Rescale the plot (y-axis) and put the breaking point in the middle of the y-axis
ManhattanPlot(
    simdata, pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    title = "Simulated P.Values - Significant", rescale = TRUE, signif_rel_pos = 0.5)
}

sig <- simdata$P.value < 5e-07

simdata$label <- ""
simdata$label[sig] <- sprintf("Label: %i", 1:sum(sig))
simdata$label2 <- ""
i <- (simdata$chromosome == 5) & (simdata$P.value < 5e-8)
simdata$label2[i] <- paste("Chromosome 5 label", 1:sum(i))

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Label the points with labels
ManhattanPlot(simdata, label_by = "label", pval_by = "P.value", chr_by = "chromosome",
    pos_by = "position", title = "Simulated P.Values with labels", label_size = 4)
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Label the points with labels and use a different color for the labels
ManhattanPlot(simdata, label_by = "label2", pval_by = "P.value", chr_by = "chromosome",
    pos_by = "position", title = "Simulated P.Values with labels",
    label_size = 3, label_fg = "black")
}

simdata$color <- "Not Significant"
simdata$color[simdata$P.value <= 5e-8] <- "Significant"

if (requireNamespace("ggmanh", quietly = TRUE)) {
```

```
# Highlight points with shapes
ManhattanPlot(simdata, title = "Highlight Points with shapes",
    pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    highlight = "color == 'Significant'", highlight_color = NULL, highlight_shape = 6,
    highlight_size = 5, pt_alpha = 0.2, pt_size = 1)
}

if (requireNamespace("ggmanh", quietly = TRUE)) {
# Highlight points with colors
ManhattanPlot(simdata, title = "Highlight Points",
    pval_by = "P.value", chr_by = "chromosome", pos_by = "position",
    highlight = "color == 'Significant'", highlight_color = "black",
    pt_color = "lightblue", pt_alpha = 0.2, pt_size = 0.1)
}
```

Network *Network*

Description

Plot a network graph

Usage

```
Network(
  links,
  nodes = NULL,
  split_by = NULL,
  split_by_sep = "_",
  split_nodes = FALSE,
  from = NULL,
  from_sep = "_",
  to = NULL,
  to_sep = "_",
  node_by = NULL,
  node_by_sep = "_",
  link_weight_by = 2,
  link_weight_name = NULL,
  link_type_by = "solid",
  link_type_name = NULL,
  node_size_by = 15,
  node_size_name = NULL,
  node_color_by = "black",
  node_color_name = NULL,
  node_shape_by = 21,
  node_shape_name = NULL,
  node_fill_by = "grey20",
```

```
  node_fill_name = NULL,
  link_alpha = 1,
  node_alpha = 0.95,
  node_stroke = 1.5,
  cluster_scale = c("fill", "color", "shape"),
  node_size_range = c(5, 20),
  link_weight_range = c(0.5, 5),
  link_arrow_offset = 20,
  link_curvature = 0,
  link_color_by = "from",
  link_color_name = NULL,
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  link_palette = ifelse(link_color_by %in% c("from", "to"), palette, "Set1"),
  link_palcolor = if (link_color_by %in% c("from", "to")) palcolor else NULL,
  directed = TRUE,
  layout = "circle",
  cluster = "none",
  add_mark = FALSE,
  mark_expand = ggplot2::unit(10, "mm"),
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  add_label = TRUE,
  label_size = 3,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  arrow = ggplot2::arrow(type = "closed", length = ggplot2::unit(0.1, "inches")),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 1,
  theme = "theme_this",
  theme_args = list(),
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
```

```
  ...
)
```

Arguments

links	A data frame containing the links between nodes.
nodes	A data frame containing the nodes. This is optional. The names of the nodes are extracted from the links data frame. If "@nodes" is provided, the nodes data frame will be extracted from the attribute nodes of the links data frame.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
split_nodes	A logical value specifying whether to split the nodes data. If TRUE, the nodes data will also be split by the split_by column.
from	A character string specifying the column name of the links data frame for the source nodes. Default is the first column of the links data frame.
from_sep	A character string to concatenate the columns in from, if multiple columns are provided.
to	A character string specifying the column name of the links data frame for the target nodes. Default is the second column of the links data frame.
to_sep	A character string to concatenate the columns in to, if multiple columns are provided.
node_by	A character string specifying the column name of the nodes data frame for the node names. Default is the first column of the nodes data frame.
node_by_sep	A character string to concatenate the columns in node_by, if multiple columns are provided.
link_weight_by	A numeric value or a character string specifying the column name of the links data frame for the link weight. If a numeric value is provided, all links will have the same weight. This determines the width of the links.
link_weight_name	A character string specifying the name of the link weight in the legend.
link_type_by	A character string specifying the type of the links. This can be "solid", "dashed", "dotted", or a column name from the links data frame. It has higher priority when it is a column name.
link_type_name	A character string specifying the name of the link type in the legend.
node_size_by	A numeric value or a character string specifying the column name of the nodes data frame for the node size. If a numeric value is provided, all nodes will have the same size.
node_size_name	A character string specifying the name of the node size in the legend.
node_color_by	A character string specifying the color of the nodes. This can be a color name, a hex code, or a column name from the nodes data frame. It has higher priority when it is a column name.
node_color_name	A character string specifying the name of the node color in the legend.

node_shape_by A numeric value or a character string specifying the column name of the nodes data frame for the node shape. If a numeric value is provided, all nodes will have the same shape.

node_shape_name
A character string specifying the name of the node shape in the legend.

node_fill_by A character string specifying the fill color of the nodes. This can be a color name, a hex code, or a column name from the nodes data frame. It has higher priority when it is a column name.

node_fill_name A character string specifying the name of the node fill in the legend.

link_alpha A numeric value specifying the transparency of the links.

node_alpha A numeric value specifying the transparency of the nodes. It only works when the nodes are filled.

node_stroke A numeric value specifying the stroke of the nodes.

cluster_scale A character string specifying how to scale the clusters. It can be "fill", "color", or "shape".

node_size_range
A numeric vector specifying the range of the node size.

link_weight_range
A numeric vector specifying the range of the link weight.

link_arrow_offset
A numeric value specifying the offset of the link arrows. So that they won't overlap with the nodes.

link_curvature A numeric value specifying the curvature of the links.

link_color_by A character string specifying the colors of the link. It can be:

- "from" means the color of the link is determined by the source node.
- "to" means the color of the link is determined by the target node.
- Otherwise, the color of the link is determined by the column name from the links data frame.

link_color_name
A character string specifying the name of the link color in the legend. Only used when link_color_by is a column name.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

link_palette A character string specifying the palette of the links. When link_color_by is "from" or "to", the palette of the links defaults to the palette of the nodes.

link_palcolor A character vector specifying the colors of the link palette. When link_color_by is "from" or "to", the colors of the link palette defaults to the colors of the node palette.

directed	A logical value specifying whether the graph is directed.
layout	A character string specifying the layout of the graph. It can be "circle", "tree", "grid", or a layout function from igraph.
cluster	A character string specifying the clustering method. It can be "none", "fast_greedy", "walktrap", "edge_betweenness", "infomap", or a clustering function from igraph.
add_mark	A logical value specifying whether to add mark for the clusters to the plot.
mark_expand	A unit value specifying the expansion of the mark.
mark_type	A character string specifying the type of the mark. It can be "hull", "ellipse", "rect", "circle", or a mark function from ggforce.
mark_alpha	A numeric value specifying the transparency of the mark.
mark_linetype	A numeric value specifying the line type of the mark.
add_label	A logical value specifying whether to add label to the nodes to the plot.
label_size	A numeric value specifying the size of the label.
label_fg	A character string specifying the foreground color of the label.
label_bg	A character string specifying the background color of the label.
label_bg_r	A numeric value specifying the background ratio of the label.
arrow	An arrow object for the links.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
legend.position	A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
axes	A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.

- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
actors <- data.frame(
  name = c("Alice", "Bob", "Cecil", "David", "Esmeralda"),
  age = c(48, 33, 45, 34, 21),
  shape = c(21, 22, 21, 22, 23),
  gender = c("F", "M", "F", "M", "F")
)
relations <- data.frame(
  from = c("Bob", "Cecil", "Cecil", "David", "David", "Esmeralda", "Bob", "Alice",
     "Cecil", "David"),
  to = c("Alice", "Bob", "Alice", "Alice", "Bob", "Alice", "Bob", "Alice", "Cecil",
     "David"),
  friendship = c(4, 5, 5, 2, 1, 1, 2, 1, 3, 4),
  type = c(1, 1, 1, 1, 1, 2, 2, 2, 2, 2)
)
Network(relations, actors)
Network(relations, actors, theme = "theme_blank", theme_args = list(add_coord = FALSE))
Network(relations, actors, link_weight_by = "friendship", node_size_by = "age",
```

```
 link_weight_name = "FRIENDSHIP", node_fill_by = "gender", link_color_by = "to",
 link_type_by = "type", node_color_by = "black", layout = "circle", link_curvature = 0.2)
Network(relations, actors, layout = "tree", directed = FALSE, cluster = "fast_greedy",
 add_mark = TRUE)
Network(relations, actors, split_by = "type")
```

---

palette_list *A list of palettes for use in data visualization*

Description

A list of palettes for use in data visualization

Examples

```
## Not run:
if (interactive()) {
  library(stringr)
  library(RColorBrewer)
  library(Redmonder)
  library(rcartocolor)
  library(nord)
  library(viridis)
  library(pals)
  library(dichromat)
  library(jcolors)
  library(scales)
  library(ggthemes)
  syspals <- utils::getFromNamespace("syspals", "pals")
  brewer.pal.info <- RColorBrewer::brewer.pal.info
  ggsci_db <- utils::getFromNamespace("ggsci_db", "ggsci")
  redmonder.pal.info <- Redmonder::redmonder.pal.info
  metacartocolors <- rcartocolor::metacartocolors
  rownames(metacartocolors) <- metacartocolors$Name
  nord_palettes <- nord::nord_palettes
  viridis_names <- c("magma", "inferno", "plasma", "viridis", "cividis", "rocket",
     "mako", "turbo")
  viridis_palettes <- lapply(stats::setNames(viridis_names, viridis_names),
     function(x) viridis::viridis(100, option = x))
  ocean_names <- names(syspals)[grep("ocean", names(syspals))]
  ocean_palettes <- syspals[ocean_names]
  dichromat_palettes <- dichromat::colorschemes
  jcolors_names <- paste0("jcolors-", c("default", "pal2", "pal3", "pal4", "pal5",
     "pal6", "pal7", "pal8", "pal9", "pal10", "pal11", "pal12", "rainbow"))
  custom_names <- c("jet", "simspec", "GdRd")
  custom_palettes <- list(
    oompaBase::jetColors(N = 100),
    c("#c22b86", "#f769a1", "#fcc5c1", "#253777", "#1d92c0", "#9ec9e1", "#015b33",
        "#42aa5e", "#d9f0a2", "#E66F00", "#f18c28", "#FFBB61"),
```

```
  c("gold", "red3")
)
names(custom_palettes) <- custom_names
seurat_discrete_palettes <- list(
   alphabet = c(
     "#F0A0FF", "#0075DC", "#993F00", "#4C005C", "#191919", "#005C31",
     "#2BCE48", "#FFCC99", "#808080", "#94FFB5", "#8F7C00", "#9DCC00",
     "#C20088", "#003380", "#FFA405", "#FFA8BB", "#426600", "#FF0010",
     "#5EF1F2", "#00998F", "#E0FF66", "#740AFF", "#990000", "#FFFF80",
     "#FFE100", "#FF5005"
   ),
   alphabet2 = c(
     "#AA0DFE", "#3283FE", "#85660D", "#782AB6", "#565656", "#1C8356",
     "#16FF32", "#F7E1A0", "#E2E2E2", "#1CBE4F", "#C4451C", "#DEA0FD",
     "#FE00FA", "#325A9B", "#FEAF16", "#F8A19F", "#90AD1C", "#F6222E",
     "#1CFFCE", "#2ED9FF", "#B10DA1", "#C075A6", "#FC1CBF", "#B00068",
     "#FBE426", "#FA0087"
   ),
   glasbey = c(
     "#0000FF", "#FF0000", "#00FF00", "#000033", "#FF00B6", "#005300",
     "#FFD300", "#009FFF", "#9A4D42", "#00FFBE", "#783FC1", "#1F9698",
     "#FFACFD", "#B1CC71", "#F1085C", "#FE8F42", "#DD00FF", "#201A01",
     "#720055", "#766C95", "#02AD24", "#C8FF00", "#886C00", "#FFB79F",
     "#858567", "#A10300", "#14F9FF", "#00479E", "#DC5E93", "#93D4FF",
     "#004CFF", "#F2F318"
   ),
   polychrome = c(
     "#5A5156", "#E4E1E3", "#F6222E", "#FE00FA", "#16FF32", "#3283FE",
     "#FEAF16", "#B00068", "#1CFFCE", "#90AD1C", "#2ED9FF", "#DEA0FD",
     "#AA0DFE", "#F8A19F", "#325A9B", "#C4451C", "#1C8356", "#85660D",
     "#B10DA1", "#FBE426", "#1CBE4F", "#FA0087", "#FC1CBF", "#F7E1A0",
     "#C075A6", "#782AB6", "#AAF400", "#BDCDFF", "#822E1C", "#B5EFB5",
     "#7ED7D1", "#1C7F93", "#D85FF7", "#683B79", "#66B0FF", "#3B00FB"
   ),
   stepped = c(
     "#990F26", "#B33E52", "#CC7A88", "#E6B8BF", "#99600F", "#B3823E",
     "#CCAA7A", "#E6D2B8", "#54990F", "#78B33E", "#A3CC7A", "#CFE6B8",
     "#0F8299", "#3E9FB3", "#7ABECC", "#B8DEE6", "#3D0F99", "#653EB3",
     "#967ACC", "#C7B8E6", "#333333", "#666666", "#999999", "#CCCCCC"
   ),
   parade = c(
     '#ff6969', '#9b37ff', '#cd3737', '#69cdff', '#ffff69', '#69cdcd',
     '#9b379b', '#3737cd', '#ffff9b', '#cdff69', '#ff9b37', '#37ffff',
     '#9b69ff', '#37cd69', '#ff3769', '#ff3737', '#37ff9b', '#cdcd37',
     '#3769cd', '#37cdff', '#9b3737', '#ff699b', '#9b9bff', '#cd9b37',
     '#69ff37', '#cd3769', '#cd69cd', '#cd6937', '#3737ff', '#cdcd69',
     '#ff9b69', '#cd37cd', '#9bff37', '#cd379b', '#cd6969', '#69ff9b',
     '#ff379b', '#9bff9b', '#6937ff', '#69cd37', '#cdff37', '#9bff69',
     '#9b37cd', '#ff37ff', '#ff37cd', '#ffff37', '#37cd9b', '#379bff',
     '#ffcd37', '#379b37', '#ff9bff', '#379b9b', '#69ffcd', '#379bcd',
     '#ff69ff', '#ff9b9b', '#37ff69', '#ff6937', '#6969ff', '#699bff',
     '#ffcd69', '#69ffff', '#37ff37', '#6937cd', '#37cd37', '#3769ff',
     '#cd69ff', '#6969cd', '#9bcd37', '#69ff69', '#37cdcd', '#cd37ff',
```

```
    '#37379b', '#37ffcd', '#69cd69', '#ff69cd', '#9bffff', '#9b9b37'
  )
)
seurat_continuous_palettes <- list(
  seurat = hue_pal()(16),
  seurat.16 = hue_pal()(16),
  seurat.32 = hue_pal()(32),
  seurat.64 = hue_pal()(64)
)
stripe_palettes <- list(
  stripe = rep(c("white", "grey60"), 8),
  stripe.16 = rep(c("white", "grey60"), 8),
  stripe.32 = rep(c("white", "grey60"), 16),
  stripe.64 = rep(c("white", "grey60"), 32)
)
tableau_palettes <- list()
orig_tableau_palettes <- ggthemes::ggthemes_data[["tableau"]][["color-palettes"]]
for (g in names(orig_tableau_palettes)) {
  for (pal in names(orig_tableau_palettes[[g]])) {
    palcolors <- as.list(orig_tableau_palettes[[g]][[pal]])
    if (!is.null(palcolors$name)) {
      tableau_palettes[[pal]] <- stats::setNames(palcolors$value, palcolors$name)
    } else {
      tableau_palettes[[pal]] <- palcolors$value
    }
  }
}

palette_list <- list()
all_colors <- c(
  rownames(brewer.pal.info), names(ggsci_db), rownames(redmonder.pal.info),
  rownames(metacartocolors), names(nord_palettes), names(viridis_palettes),
 ocean_names, names(dichromat_palettes), jcolors_names, names(seurat_discrete_palettes),
  names(seurat_continuous_palettes), custom_names, names(stripe_palettes),
  names(tableau_palettes)
)
for (pal in all_colors) {
  if (!pal %in% all_colors) {
    stop(paste0("Invalid pal Must be one of ", paste0(all_colors, collapse = ",")))
  }
  if (pal %in% rownames(brewer.pal.info)) {
    pal_n <- brewer.pal.info[pal, "maxcolors"]
    pal_category <- brewer.pal.info[pal, "category"]
    if (pal_category == "div") {
      palcolor <- rev(brewer.pal(name = pal, n = pal_n))
    } else {
      if (pal == "Paired") {
        palcolor <- brewer.pal(12, "Paired")[c(1:4, 7, 8, 5, 6, 9, 10, 11, 12)]
      } else {
        palcolor <- brewer.pal(name = pal, n = pal_n)
      }
    }
    if (pal_category == "qual") {
```

```
    attr(palcolor, "type") <- "discrete"
  } else {
    attr(palcolor, "type") <- "continuous"
  }
} else if (pal %in% names(ggsci_db)) {
  if (pal %in% c("d3", "uchicago", "material")) {
    for (subpal in names(ggsci_db[[pal]])) {
      palcolor <- ggsci_db[[pal]][[subpal]]
      if (pal == "material") {
        attr(palcolor, "type") <- "continuous"
      } else {
        attr(palcolor, "type") <- "discrete"
      }
      palette_list[[paste0(pal, "-", subpal)]] <- palcolor
    }
    next
  } else {
    palcolor <- ggsci_db[[pal]][[1]]
    if (pal == "gsea") {
      attr(palcolor, "type") <- "continuous"
    } else {
      attr(palcolor, "type") <- "discrete"
    }
  }
} else if (pal %in% rownames(redmonder.pal.info)) {
  pal_n <- redmonder.pal.info[pal, "maxcolors"]
  pal_category <- redmonder.pal.info[pal, "category"]
  if (pal_category == "div") {
    palcolor <- rev(redmonder.pal(name = pal, n = pal_n))
  } else {
    palcolor <- redmonder.pal(name = pal, n = pal_n)
  }
  if (pal_category == "qual") {
    attr(palcolor, "type") <- "discrete"
  } else {
    attr(palcolor, "type") <- "continuous"
  }
} else if (pal %in% rownames(metacartocolors)) {
  pal_n <- metacartocolors[pal, "Max_n"]
  palcolor <- carto_pal(name = pal, n = pal_n)
  if (pal_category == "qualitative") {
    attr(palcolor, "type") <- "discrete"
  } else {
    attr(palcolor, "type") <- "continuous"
  }
} else if (pal %in% names(nord_palettes)) {
  palcolor <- nord_palettes[[pal]]
  attr(palcolor, "type") <- "discrete"
} else if (pal %in% names(viridis_palettes)) {
  palcolor <- viridis_palettes[[pal]]
  attr(palcolor, "type") <- "continuous"
} else if (pal %in% names(ocean_palettes)) {
  palcolor <- ocean_palettes[[pal]]
```

```
      attr(palcolor, "type") <- "continuous"
    } else if (pal %in% names(dichromat_palettes)) {
      palcolor <- dichromat_palettes[[pal]]
      if (pal %in% c("Categorical.12", "SteppedSequential.5")) {
        attr(palcolor, "type") <- "discrete"
      } else {
        attr(palcolor, "type") <- "continuous"
      }
    } else if (pal %in% jcolors_names) {
      palcolor <- jcolors(palette = gsub("jcolors-", "", pal))
      if (pal %in% paste0("jcolors-", c("pal10", "pal11", "pal12", "rainbow"))) {
        attr(palcolor, "type") <- "continuous"
      } else {
        attr(palcolor, "type") <- "discrete"
      }
    } else if (pal %in% custom_names) {
      palcolor <- custom_palettes[[pal]]
      if (pal %in% c("jet")) {
        attr(palcolor, "type") <- "continuous"
      } else {
        attr(palcolor, "type") <- "discrete"
      }
    } else if (pal %in% names(seurat_discrete_palettes)) {
      palcolor <- seurat_discrete_palettes[[pal]]
      attr(palcolor, "type") <- "discrete"
    } else if (pal %in% names(seurat_continuous_palettes)) {
      palcolor <- seurat_continuous_palettes[[pal]]
      attr(palcolor, "type") <- "continuous"
    } else if (pal %in% names(stripe_palettes)) {
      palcolor <- stripe_palettes[[pal]]
      attr(palcolor, "type") <- "discrete"
    } else if (pal %in% names(tableau_palettes)) {
      palcolor <- tableau_palettes[[pal]]
      attr(palcolor, "type") <- "discrete"
    }
    palette_list[[pal]] <- palcolor
  }
}

## End(Not run)
```

---

palette_this *Color palettes collected in plotthis.*

Description

Color palettes collected in plotthis.

Usage

```
palette_this(
  x,
  n = 100,
  palette = "Paired",
  palcolor = NULL,
  type = "auto",
  keep_names = TRUE,
  alpha = 1,
  matched = FALSE,
  reverse = FALSE,
  NA_keep = FALSE,
  NA_color = "grey80",
  transparent = TRUE
)
```

Arguments

x	A vector of character/factor or numeric values. If missing, numeric values 1:n will be used as x.
n	The number of colors to return for numeric values.
palette	Palette name. All available palette names can be queried with `show_palettes()`.
palcolor	Custom colors used to create a color palette.
type	Type of `x`. Can be one of "auto", "discrete" or "continuous". The default is "auto", which automatically detects if `x` is a numeric value.
keep_names	Whether to keep the names of the color vector.
alpha	The alpha value of the colors. Default is 1.
matched	If `TRUE`, will return a color vector of the same length as `x`.
reverse	Whether to invert the colors.
NA_keep	Whether to keep the color assignment to NA in `x`.
NA_color	Color assigned to NA if NA_keep is `TRUE`.
transparent	Whether to make the colors transparent when alpha < 1. When `TRUE`, `ggplot2::alpha()` is used to make the colors transparent. Otherwise, `adjcolors` is used to adjust the colors based on the alpha. The color will be not be actually transparent. For example, `ggplot2::alpha("red", 0.5) == "#FF000080"`; while `adjcolors("red", 0.5) == "#FF8080"`.

Value

A vector of colors.

PieChart *Pie Chart*

Description

Pie chart to illustrate numerical proportion of each group.

Usage

```
PieChart(
  data,
  x,
  y = NULL,
  label = y,
  split_by = NULL,
  split_by_sep = "_",
  clockwise = TRUE,
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  aspect.ratio = 1,
  keep_na = FALSE,
  keep_empty = FALSE,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
```

```
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string of the column name to plot on the y-axis. A numeric column is expected. If NULL, the count of each x column will be used.
label	Which column to use as the label. NULL means no label. Default is the same as y. If y is NULL, you should use ".y" to specify the count as the label. If TRUE, the y values will be used as the label.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See `split_by`
clockwise	A logical value to draw the pie chart clockwise or not.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
alpha	A numeric value specifying the transparency of the plot.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
keep_na	A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be

one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

legend.position
: A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

seed
: The random seed to use. Default is 8525.

axes
: A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
   x = factor(c("A", "B", "C", NA, "E", "F", "G", "H"), levels = LETTERS[1:8]),
   y = c(10, 8, 16, 4, 6, 12, 14, 2),
   group = factor(c("G1", "G1", NA, NA, "G3", "G3", "G4", "G4"),
       levels = c("G1", "G2", "G3", "G4")),
   facet = factor(c("F1", NA, "F3", "F4", "F1", NA, "F3", "F4"),
       levels = c("F1", "F2", "F3", "F4"))
)
PieChart(data, x = "x", y = "y")
PieChart(data, x = "x", y = "y", keep_na = TRUE, keep_empty = TRUE)
PieChart(data, x = "x", y = "y", clockwise = FALSE)
PieChart(data, x = "x", y = "y", clockwise = FALSE,
         keep_na = TRUE, keep_empty = TRUE)
PieChart(data, x = "x", y = "y", label = "group")
PieChart(data, x = "x", y = "y", facet_by = "facet")
PieChart(data, x = "x", y = "y", facet_by = c("facet", "group"),
    keep_empty = 'level')
PieChart(data, x = "x", y = "y", facet_by = c("facet", "group"),
    keep_empty = TRUE)
PieChart(data, x = "x", y = "y", split_by = "group")
PieChart(data, x = "x", y = "y", split_by = "group",
         palette = list(G1 = "Reds", G2 = "Blues", G3 = "Greens", G4 = "Purp"))

# y from count
PieChart(data, x = "group")
# add label
PieChart(data, x = "group", label = ".y")  # or label = TRUE
```

QQPlot *QQ plot*

Description

QQ plot is a graphical tool to compare two distributions by plotting their quantiles against each other.

Usage

```
QQPlot(
  data,
  val,
  val_trans = NULL,
  type = c("qq", "pp"),
  split_by = NULL,
  split_by_sep = "_",
  band = NULL,
  line = list(),
  point = list(),
  fill_name = "Bands",
  band_alpha = 0.5,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlim = NULL,
  ylim = NULL,
  xlab = ifelse(type == "qq", "Theoretical Quantiles", "Probability Points"),
  ylab = ifelse(type == "qq", "Sample Quantiles", "Cumulative Probability"),
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
```

```
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
val	A character string of the column name for the values to plot. A numeric column is expected.
val_trans	A function to transform the values before plotting. Default is NULL, which means no transformation.
type	A character string to specify the type of plot. Default is "qq", which means QQ plot. Other options are "pp", which means PP plot.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
band	A list of arguments to pass to qqplotr::stat_qq_band() or qqplotr::stat_pp_band(), depending on the value of type. Default is NULL, which means no band. If an empty list or TRUE is provided, the default arguments will be used. Multiple bands can be added by providing a list of lists.
line	A list of arguments to pass to qqplotr::stat_qq_line() or qqplotr::stat_pp_line(), depending on the value of type. Default is list(), which means to add a line with default arguments. If NULL is provided, no line will be added.
point	A list of arguments to pass to qqplotr::stat_qq_point() or qqplotr::stat_pp_point(), depending on the value of type. Default is list(), which means to add points with default arguments. If NULL is provided, no points will be added (not recommended).
fill_name	A character string to name the legend of fill. Default is "Band Type".
band_alpha	A numeric value to set the alpha of all bands. Default is 0.5. It is a shortcut for setting alpha of all bands. You can override it by setting alpha in band argument. For example, band = list(list(alpha = 0.3), list(alpha = 0.7)).
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales
: Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlim
: A numeric vector of length 2 to set the x-axis limits.

ylim
: A numeric vector of length 2 to set the y-axis limits.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

seed
: The random seed to use. Default is 8525.

axes
: A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
  - 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides
: A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
data <- data.frame(norm = rnorm(100))

QQPlot(data, val = "norm", band = TRUE)
QQPlot(data, val = "norm", band = list(
    list(bandType = "ks", mapping = ggplot2::aes(fill = "KS"), alpha = 0.3),
    list(bandType = "ts", mapping = ggplot2::aes(fill = "TS")),
    list(bandType = "pointwise", mapping = ggplot2::aes(fill = "Normal")),
    list(bandType = "boot", mapping = ggplot2::aes(fill = "Bootstrap"))
), band_alpha = 0.6)

data(airquality, package = "datasets")
di <- "exp" # exponential distribution
dp <- list(rate = 2) # exponential rate parameter
QQPlot(airquality, val = "Ozone",
    band = list(distribution = di, dparams = dp),
    line = list(distribution = di, dparams = dp),
    point = list(distribution = di, dparams = dp)
)

de <- TRUE # enabling the detrend option
QQPlot(airquality, val = "Ozone",
    band = list(distribution = di, dparams = dp, detrend = de),
    line = list(distribution = di, dparams = dp, detrend = de),
    point = list(distribution = di, dparams = dp, detrend = de)
)

QQPlot(data, val = "norm", type = "pp", band = TRUE)

dp <- list(mean = 2, sd = 2) # shifted and rescaled Normal parameters
QQPlot(data, val = "norm", type = "pp",
    band = list(dparams = dp),
    point = list(dparams = dp))
```

```
QQPlot(data, val = "norm", type = "pp", band = TRUE,
    line = list(ab = c(.2, .5)))

di <- "exp"
dp <- list(rate = .022) # value is based on some empirical tests
de <- TRUE
QQPlot(airquality, val = "Ozone", type = "pp",
   band = list(distribution = di, detrend = de, dparams = dp),
   line = list(detrend = de),
   point = list(distribution = di, detrend = de, dparams = dp),
   ylim = c(-.5, .5)
)
```

RadarPlot *Radar plot / Spider plot*

Description

Create a radar plot or spider plot for a series of data. Radar plot uses circles as the plot grid and Spider plot uses polygons.

Usage

```
RadarPlot(
  data,
  x,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  y = NULL,
  group_name = NULL,
  groups = NULL,
  scale_y = c("group", "global", "x", "none"),
  y_min = 0,
  y_max = NULL,
  y_nbreaks = 4,
  bg_color = "grey80",
  bg_alpha = 0.1,
  fill = TRUE,
  linewidth = 1,
  pt_size = 4,
  max_charwidth = 16,
  split_by = NULL,
  split_by_sep = "_",
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
```

```
  palcolor = NULL,
  palreverse = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  alpha = 0.2,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  keep_na = FALSE,
  keep_empty = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

SpiderPlot(
  data,
  x,
  x_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  y = NULL,
  group_name = NULL,
  groups = NULL,
  scale_y = c("group", "global", "x", "none"),
  y_min = 0,
  y_max = NULL,
  y_nbreaks = 4,
  bg_color = "grey80",
  bg_alpha = 0.1,
  fill = TRUE,
  linewidth = 1,
  pt_size = 4,
  max_charwidth = 16,
  split_by = NULL,
  split_by_sep = "_",
```

```
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  alpha = 0.2,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  keep_na = FALSE,
  keep_empty = FALSE,
  title = NULL,
  subtitle = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
y	A character string specifying the column name of the data frame to plot for the y-axis.
group_name	A character string to name the legend of group.
groups	A character vector of group values (in the group_by column) to include in the plot. If NULL, all groups will be included. This can be used to exclude certain

groups from the plot or to specify the order of groups in the legend. Only applicable when group_by is provided. And this implies keep_empty for group_by is FALSE, which means the groups not in the data will not be shown in the legend.

scale_y — How should the y-axis be scaled? Default is "group". Other options are "global", "x" and "none".

- If "group", the y-axis will be scaled to the fraction within each group.
- If "global", the y-axis will be scaled to the fraction of the total.
- If "x", the y-axis will be scaled to the fraction of the total within each x-axis group.
- If "none", the y-axis will be scaled to the count of each x-axis group.

y_min — A numeric value to set the minimum value of the y-axis.

y_max — A numeric value to set the maximum value of the y-axis.

y_nbreaks — A numeric value to set the number of breaks in the y-axis.

bg_color — A character string to set the background color of the plot.

bg_alpha — A numeric value to set the transparency of the background color.

fill — A logical value to fill the polygons with colors.

linewidth — A numeric value to set the width of the lines.

pt_size — A numeric value to set the size of the points.

max_charwidth — A numeric value to set the maximum character width for the x labels.

split_by — The column(s) to split data by and plot separately.

split_by_sep — The separator for multiple split_by columns. See split_by

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

alpha
: A numeric value specifying the transparency of the plot.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

  - `FALSE` (default): Drop empty factor levels from the data before plotting.
  - `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
  - `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

seed
: The random seed to use. Default is 8525.

combine
: Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow
: A numeric value specifying the number of rows in the facet.

ncol
: A numeric value specifying the number of columns in the facet.

byrow
: A logical value indicating whether to fill the plots by row.

axes
: A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

  - 'keep' will retain all axes in individual plots.
  - 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
  - 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles
: A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'keep' will retain all axis titles in individual plots.
  - 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
  - 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides
: A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:
  - 'collect' will collect guides below to the given nesting level, removing duplicates.
  - 'keep' will stop collection at this level and let guides be placed alongside their plot.
  - 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design
: Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

...
: Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
# use the count
data <- data.frame(
   x = factor(
    c(rep("A", 20), rep("B", 30), rep(NA, 30), rep("D", 40), rep("E", 50)),
    levels = LETTERS[1:5]
   ),
   group = factor(
    sample(c("G1", NA, "G3", "G4"), 170, replace = TRUE),
    levels = c("G1", "G2", "G3", "G4")
   )
)

RadarPlot(data, x = "x")
RadarPlot(data, x = "x", keep_na = TRUE, keep_empty = TRUE)
RadarPlot(data, x = "x", bg_color = "lightpink")
RadarPlot(data, x = "x", scale_y = "none")
RadarPlot(data, x = "x", group_by = "group", keep_na = TRUE)
RadarPlot(data, x = "x", facet_by = "group")
SpiderPlot(data, x = "x")
SpiderPlot(data, x = "x", group_by = "group")
# use the y value
```

```
data <- data.frame(
   x = rep(LETTERS[1:5], 2),
   y = c(1, 3, 6, 4, 2, 5, 7, 8, 9, 10),
   group = rep(c("G1", "G2"), each = 5)
)
RadarPlot(data, x = "x", y = "y", scale_y = "none", group_by = "group")
RadarPlot(data, x = "x", y = "y", facet_by = "group")
RadarPlot(data, x = "x", y = "y", split_by = "group")
RadarPlot(data, x = "x", y = "y", split_by = "group",
          palette = c(G1 = "Set1", G2 = "Paired"))
```

RarefactionPlot *RarefactionPlot*

Description

This function generates a rarefraction plot for a given dataset.

Usage

```
RarefactionPlot(
  data,
  type = 1,
  se = NULL,
  group_by = "group",
  group_by_sep = "_",
  group_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 0.2,
  pt_size = 3,
  line_width = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
```

```
  xlab = NULL,
  ylab = NULL,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
type	three types of plots: sample-size-based rarefaction/extrapolation curve (type = 1); sample completeness curve (type = 2); coverage-based rarefaction/extrapolation curve (type = 3).
se	a logical variable to display confidence interval around the estimated sampling curve. Default to NULL which means TRUE if the data has the lower and upper bounds.
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	A character string indicating how to separate the group_by column if both "q" and "group" are used. for 'group_by'. Default to "_".
group_name	A character string indicating the name of the group, showing as the legend title.
split_by	A character string indicating how to split the data and plots Possible values are "q" and "group"
split_by_sep	The separator for multiple split_by columns. See split_by
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
alpha	A numeric value specifying the transparency of the plot.
pt_size	A numeric value specifying the size of the points.

line_width A numeric value specifying the width of the lines.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

seed The random seed to use. Default is 8525.

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow A logical value indicating whether to fill the plots by row.

axes A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
spider <- list(
   Girdled = c(46, 22, 17, 15, 15, 9, 8, 6, 6, 4, rep(2, 4), rep(1, 12)),
   Logged = c(88, 22, 16, 15, 13, 10, 8, 8, 7, 7, 7, 5, 4, 4, 4, 3, 3, 3, 3,
     2, 2, 2, 2, rep(1, 14))
)

RarefactionPlot(spider)
RarefactionPlot(spider, q = c(0, 1, 2), facet_by = "q")
RarefactionPlot(spider, q = c(0, 1, 2), split_by = "q")
RarefactionPlot(spider, q = c(0, 1, 2), split_by = "q",
                palette = c("0" = "Paired", "1" = "Set1", "2" = "Dark2"))
RarefactionPlot(spider, q = c(0, 1, 2), group_by = "q",
 facet_by = "group", palette = "Set1", type = 3)
```

RidgePlot *Ridge Plot*

Description

Ridge plot to illustrate the distribution of the data in different groups.

Usage

```
RidgePlot(
  data,
  x = NULL,
  in_form = c("long", "wide"),
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  scale = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  add_vline = NULL,
  vline_type = "solid",
  vline_color = TRUE,
  vline_width = 0.5,
  vline_alpha = 1,
  flip = FALSE,
  alpha = 0.8,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  x_text_angle = 90,
  reverse = FALSE,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "none",
  legend.direction = "vertical",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
```

```
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
in_form	A character string specifying the form of the data. Default is "long".
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of 'group_by', if 'legend.position' is not "none".
scale	A numeric value to scale the ridges. See also geom_density_ridges.
keep_na	A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.
keep_empty	One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc. • FALSE (default): Drop empty factor levels from the data before plotting. • TRUE: Keep empty factor levels and show them as a separate category in the plot. • "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels
add_vline	A numeric vector or a named list of numeric values to add vertical lines to the plot. If a named list is provided, the names should match the levels of 'group_by'. If TRUE, the vertical lines will be added at the mean of each group.
vline_type	The type of line to draw for the vertical line.
vline_color	The color of the vertical line. If TRUE, the vertical lines will be colored according to the group colors.
vline_width	The width of the vertical line.

vline_alpha The alpha value of the vertical line.

flip A logical value. If TRUE, the plot will be flipped.

alpha A numeric value specifying the transparency of the plot.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

x_text_angle A numeric value specifying the angle of the x-axis text.

reverse A logical value. If TRUE, reverse the order of the groups on the y-axis.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow A logical value indicating whether to fill the plots by row.

seed — The random seed to use. Default is 8525.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects. If no split_by is provided, a single plot (ggplot object) will be returned. If 'combine' is TRUE, a wrap_plots object will be returned. If 'combine' is FALSE, a list of ggplot objects will be returned.

Examples

```
set.seed(8525)
data <- data.frame(
   x = c(rnorm(250, -1), rnorm(250, 1)),
   group = factor(rep(c("A", NA, LETTERS[3:5]), each = 100), levels = LETTERS[1:6])
)
RidgePlot(data, x = "x")  # fallback to a density plot
RidgePlot(data, x = "x", add_vline = 0, vline_color = "black")
RidgePlot(data, x = "x", group_by = "group")
RidgePlot(data, x = "x", group_by = "group",
```

```
   keep_na = TRUE, keep_empty = TRUE)
RidgePlot(data, x = "x", group_by = "group", reverse = TRUE)
RidgePlot(data, x = "x", group_by = "group",
   add_vline = TRUE, vline_color = TRUE, alpha = 0.7)
RidgePlot(data, x = "x", facet_by = "group",
   keep_na = TRUE, keep_empty = TRUE)

# wide form
data_wide <- data.frame(
   A = rnorm(100),
   B = rnorm(100),
   C = rnorm(100),
   D = rnorm(100),
   E = rnorm(100),
   group = sample(letters[1:4], 100, replace = TRUE)
)
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide")
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide", facet_by = "group")
RidgePlot(data_wide, group_by = LETTERS[1:5], in_form = "wide", split_by = "group",
   palette = list(a = "Reds", b = "Blues", c = "Greens", d = "Purples"))
```

RingPlot *Ring Plot*

Description

A ring plot is like pie chart but with multiple rings.

Usage

```
RingPlot(
  data,
  x = NULL,
  y = NULL,
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  label = NULL,
  split_by = NULL,
  split_by_sep = "_",
  facet_by = NULL,
  facet_scales = "free_y",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
```

```
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  aspect.ratio = 1,
  keep_na = FALSE,
  keep_empty = FALSE,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character vector specifying the column as the rings of the plot.
y	A character vector specifying the column as the y axis of the plot. Default is NULL, meaning the y axis is the count of the data.
group_by	A character vector specifying the column as the group_by of the plot. How the ring is divided.
group_by_sep	A character string to concatenate the columns in group_by, if multiple columns are provided.
group_name	A character string to specify the name of the group_by in the legend.
label	A logical value indicating whether to show the labels on the rings. The labels should be the values of group_by. Default is NULL, meaning no labels for one ring and showing the labels for multiple rings.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol
: A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow
: A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow
: A logical value indicating whether to fill the plots by row. Default is TRUE.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

palette
: A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor
: A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

alpha
: A numeric value specifying the transparency of the plot.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

keep_na
: A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty
: One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

  - `FALSE` (default): Drop empty factor levels from the data before plotting.
  - `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
  - `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
axes	A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axes in individual plots. • 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout. • 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.
axis_titles	A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axis titles in individual plots. • 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction. • 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.
guides	A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'collect' will collect guides below to the given nesting level, removing duplicates. • 'keep' will stop collection at this level and let guides be placed alongside their plot. • 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.
design	Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

See Also

PieChart

Examples

```
RingPlot(datasets::iris, group_by = "Species")
RingPlot(datasets::mtcars, x = "cyl", group_by = "carb", facet_by = "vs")
RingPlot(datasets::mtcars, x = "cyl", group_by = "carb", split_by = "vs",
        palette = c("0" = "Set1", "1" = "Paired"))

data <- data.frame(
  x = factor(c("A", "B", NA, "D", "A", "B", NA, "D"), levels = c("A", "B", "C", "D")),
  y = c(1, 2, 5, 3, 4, 5, 2, 6),
  group = factor(c("a", "a", "a", NA, NA, "c", "c", "c"), levels = c("a", "b", "c"))
)
RingPlot(data, x = "x", y = "y", group_by = "group")
RingPlot(data, x = "x", y = "y", group_by = "group",
        keep_na = TRUE, keep_empty = TRUE)
RingPlot(data, x = "x", y = "y", group_by = "group",
        keep_na = TRUE, keep_empty = list(x = FALSE, group = 'level'))
```

ROCCurve *ROC curve*

Description

A wrapped function around `plotROC` package to create ROC curves.

Usage

```
ROCCurve(
  data,
  truth_by,
  score_by,
  pos_label = NULL,
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  x_axis_reverse = FALSE,
  percent = FALSE,
  ci = NULL,
  n_cuts = 0,
  cutoffs_at = NULL,
  cutoffs_labels = NULL,
  cutoffs_accuracy = 0.001,
  cutoffs_pt_size = 5,
  cutoffs_pt_shape = 4,
  cutoffs_pt_stroke = 1,
  cutoffs_labal_fg = "black",
```

```
  cutoffs_label_size = 4,
  cutoffs_label_bg = "white",
  cutoffs_label_bg_r = 0.1,
  show_auc = c("auto", "none", "legend", "plot"),
  auc_accuracy = 0.01,
  auc_size = 4,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = ifelse(x_axis_reverse, "Specificity", "1 - Specificity"),
  ylab = "Sensitivity",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
truth_by	A character string of the column name that contains the true class labels. (a.k.a. the binary outcome, 1/0 or TRUE/FALSE.)
score_by	character strings of the column names that contains the predicted scores. When multiple columns are provided, the ROC curve is plotted for each column.
pos_label	A character string of the positive class label. When NULL, the labels will be handled by the plotROC package.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by

group_by — Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator

group_by_sep — The separator for multiple group_by columns. See group_by

group_name — A character string to name the legend of the ROC curve groups.

x_axis_reverse — A logical to reverse the x-axis, that is from 1 to 0.

percent — A logical to display the x and y axis as percentages.

ci — A list of arguments to pass to plotROC::geom_rocci() to add confidence intervals. When NULL, no confidence intervals are added.

n_cuts — An integer to specify the number of cutpoints on the ROC curve. It will be the quantiles of the predicted scores.

cutoffs_at — Vector of user supplied cutoffs to plot as points. If non-NULL, it will override the values of n_cuts and plot the observed cutoffs closest to the user-supplied ones. Both cutoffs_at and cutoffs.labels will be passed to plotROC::geom_roc(). Other than numeric values, the following special values are allowed. These values are the methods of OptimalCutpoints::optimal.cutpoints(), they are literally:

- "CB" (cost-benefit method);
- "MCT" (minimizes Misclassification Cost Term);
- "MinValueSp" (a minimum value set for Specificity);
- "MinValueSe" (a minimum value set for Sensitivity);
- "ValueSe" (a value set for Sensitivity);
- "MinValueSpSe" (a minimum value set for Specificity and Sensitivity);
- "MaxSp" (maximizes Specificity);
- "MaxSe" (maximizes Sensitivity);
- "MaxSpSe" (maximizes Sensitivity and Specificity simultaneously);
- "MaxProdSpSe" (maximizes the product of Sensitivity and Specificity or Accuracy Area);
- "ROC01" (minimizes distance between ROC plot and point (0,1));
- "SpEqualSe" (Sensitivity = Specificity);
- "Youden" (Youden Index);
- "MaxEfficiency" (maximizes Efficiency or Accuracy, similar to minimize Error Rate);
- "Minimax" (minimizes the most frequent error);
- "MaxDOR" (maximizes Diagnostic Odds Ratio);
- "MaxKappa" (maximizes Kappa Index);
- "MinValueNPV" (a minimum value set for Negative Predictive Value);
- "MinValuePPV" (a minimum value set for Positive Predictive Value);
- "ValueNPV" (a value set for Negative Predictive Value);
- "ValuePPV" (a value set for Positive Predictive Value);
- "MinValueNPVPPV" (a minimum value set for Predictive Values);
- "PROC01" (minimizes distance between PROC plot and point (0,1));

- "NPVEqualPPV" (Negative Predictive Value = Positive Predictive Value);
- "MaxNPVPPV" (maximizes Positive Predictive Value and Negative Predictive Value simultaneously);
- "MaxSumNPVPPV" (maximizes the sum of the Predictive Values);
- "MaxProdNPVPPV" (maximizes the product of Predictive Values);
- "ValueDLR.Negative" (a value set for Negative Diagnostic Likelihood Ratio);
- "ValueDLR.Positive" (a value set for Positive Diagnostic Likelihood Ratio);
- "MinPvalue" (minimizes p-value associated with the statistical Chi-squared test which measures the association between the marker and the binary result obtained on using the cutpoint);
- "ObservedPrev" (The closest value to observed prevalence);
- "MeanPrev" (The closest value to the mean of the diagnostic test values);
- "PrevalenceMatching" (The value for which predicted prevalence is practically equal to observed prevalence).

`cutoffs_labels` vector of user-supplied labels for the cutoffs. Must be a character vector of the same length as cutoffs_at.

`cutoffs_accuracy`
A numeric to specify the accuracy of the cutoff values to show.

`cutoffs_pt_size`
A numeric to specify the size of the cutoff points.

`cutoffs_pt_shape`
A numeric to specify the shape of the cutoff points.

`cutoffs_pt_stroke`
A numeric to specify the stroke of the cutoff points.

`cutoffs_labal_fg`
A character string to specify the color of the cutoff labels.

`cutoffs_label_size`
A numeric to specify the size of the cutoff labels.

`cutoffs_label_bg`
A character string to specify the background color of the cutoff labels.

`cutoffs_label_bg_r`
A numeric to specify the radius of the background of the cutoff labels.

`show_auc` A character string to specify the position of the AUC values.

- "auto" (default): Automatically determine the position based on the plot. When there is a single group or 'facet_by' is provided, the AUC is placed on the plot. Otherwise, the AUC is placed in the legend.
- "none": Do not display the AUC values.
- "legend": Display the AUC values in the legend.
- "plot": Display the AUC values on the plot (left/right bottom corner).

`auc_accuracy` A numeric to specify the accuracy of the AUC values.

`auc_size` A numeric to specify the size of the AUC values when they are displayed on the plot.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

alpha — A numeric value specifying the transparency of the plot.

facet_by — A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots

facet_scales — Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_ncol — A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow — A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

seed — The random seed to use. Default is 8525.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.

- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A `patch_work::wrap_plots` object or a list of them if `combine` is `FALSE`. You can retrieve the AUC values using `attr(p, "auc")` if `combine` is `TRUE`. If `combine` is `FALSE`, The AUC value of each plot can be retrieved using `attr(p[[i]], "auc")`.

Examples

```
set.seed(8525)

D.ex <- rbinom(200, size = 1, prob = .5)
M1 <- rnorm(200, mean = D.ex, sd = .65)
M2 <- rnorm(200, mean = D.ex, sd = 1.5)
gender <- c("Male", "Female")[rbinom(200, 1, .49) + 1]

data <- data.frame(D = D.ex, D.str = c("Healthy", "Ill")[D.ex + 1],
  gender = gender, M1 = M1, M2 = M2)

ROCCurve(data, truth_by = "D", score_by = "M1")
# will warn about the positive label
ROCCurve(data, truth_by = "D.str", score_by = "M1")
ROCCurve(data, truth_by = "D", score_by = "M1", increasing = FALSE)
# Multiple ROC curves
```

```
ROCCurve(data, truth_by = "D", score_by = c("M1", "M2"), group_name = "Method")
ROCCurve(data, truth_by = "D", score_by = "M1", group_by = "gender", show_auc = "plot")
# Reverse the x-axis and display the axes as percentages
ROCCurve(data, truth_by = "D", score_by = "M1", x_axis_reverse = TRUE, percent = TRUE)
# Pass additional arguments to geom_roc and make the curve black
ROCCurve(data, truth_by = "D", score_by = "M1", n_cuts = 10, palcolor = "black")
# Add confidence intervals
ROCCurve(data, truth_by = "D", score_by = "M1", ci = list(sig.level = .01))
# Facet by a column
ROCCurve(data, truth_by = "D", score_by = "M1", facet_by = "gender")
# Show cutoffs
ROCCurve(data, truth_by = "D", score_by = "M1", cutoffs_at = c(0, "ROC01", "SpEqualSe"))
# Split by a column
p <- ROCCurve(data, truth_by = "D", score_by = "M1", split_by = "gender",
   cutoffs_at = c(0.2, "MaxSpSe"))
p
# Retrieve the AUC values
attr(p, "auc")
# Retrieve the cutoffs
attr(p, "cutoffs")
```

SankeyPlot *Sankey / Alluvial Plot*

Description

A plot visualizing flow/movement/change from one state to another or one time to another. `AlluvialPlot` is an alias of `SankeyPlot`.

Usage

```
SankeyPlot(
  data,
  in_form = c("auto", "long", "lodes", "wide", "alluvia", "counts"),
  x,
  x_sep = "_",
  y = NULL,
  stratum = NULL,
  stratum_sep = "_",
  alluvium = NULL,
  alluvium_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  keep_empty = TRUE,
  flow = FALSE,
  expand = c(0, 0, 0, 0),
  nodes_legend = c("auto", "separate", "merge", "none"),
  nodes_color = "grey30",
  links_fill_by = NULL,
```

```
  links_fill_by_sep = "_",
  links_name = NULL,
  links_color = "gray80",
  nodes_palette = "Paired",
  nodes_palcolor = NULL,
  nodes_alpha = 1,
  nodes_label = FALSE,
  nodes_label_miny = 0,
  nodes_width = 0.25,
  links_palette = "Paired",
  links_palcolor = NULL,
  palreverse = FALSE,
  links_alpha = 0.6,
  legend.box = "vertical",
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  flip = FALSE,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)

AlluvialPlot(
  data,
  in_form = c("auto", "long", "lodes", "wide", "alluvia", "counts"),
  x,
  x_sep = "_",
  y = NULL,
```

```
    stratum = NULL,
    stratum_sep = "_",
    alluvium = NULL,
    alluvium_sep = "_",
    split_by = NULL,
    split_by_sep = "_",
    keep_empty = TRUE,
    flow = FALSE,
    expand = c(0, 0, 0, 0),
    nodes_legend = c("auto", "separate", "merge", "none"),
    nodes_color = "grey30",
    links_fill_by = NULL,
    links_fill_by_sep = "_",
    links_name = NULL,
    links_color = "gray80",
    nodes_palette = "Paired",
    nodes_palcolor = NULL,
    nodes_alpha = 1,
    nodes_label = FALSE,
    nodes_label_miny = 0,
    nodes_width = 0.25,
    links_palette = "Paired",
    links_palcolor = NULL,
    palreverse = FALSE,
    links_alpha = 0.6,
    legend.box = "vertical",
    x_text_angle = 0,
    aspect.ratio = 1,
    legend.position = "right",
    legend.direction = "vertical",
    flip = FALSE,
    theme = "theme_this",
    theme_args = list(),
    title = NULL,
    subtitle = NULL,
    xlab = NULL,
    ylab = NULL,
    facet_by = NULL,
    facet_scales = "fixed",
    facet_ncol = NULL,
    facet_nrow = NULL,
    facet_byrow = TRUE,
    seed = 8525,
    combine = TRUE,
    nrow = NULL,
    ncol = NULL,
    byrow = TRUE,
    axes = NULL,
```

```
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame in following possible formats: • "long" or "lodes": A long format with columns for x, stratum, alluvium, and y. x (required, single columns or concatenated by x_sep) is the column name to plot on the x-axis, stratum (defaults to links_fill_by) is the column name to group the nodes for each x, alluvium (required) is the column name to define the links, and y is the frequency of each x, stratum, and alluvium. • "wide" or "alluvia": A wide format with columns for x. x (required, multiple columns, x_sep won't be used) are the columns to plot on the x-axis, stratum and alluvium will be ignored. See ggalluvial::to_lodes_form for more details. • "counts": A format with counts being provides under each x. x (required, multiple columns, x_sep won't be used) are the columns to plot on the x-axis. When the first element of x is ".", values of links_fill_by (required) will be added to the plot as the first column of nodes. It is useful to show how the links are flowed from the source to the targets. • "auto" (default): Automatically determine the format based on the columns provided. When the length of x is greater than 1 and all x columns are numeric, "counts" format will be used. When the length of x is greater than 1 and ggalluvial::is_alluvia_form returns TRUE, "alluvia" format will be used. Otherwise, "lodes" format will be tried.
in_form	A character string to specify the format of the data. Possible values are "auto", "long", "lodes", "wide", "alluvia", and "counts".
x	A character string specifying the column name of the data frame to plot for the x-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
y	A character string specifying the column name of the data frame to plot for the y-axis.
stratum	A character string of the column name to group the nodes for each x. See data for more details.
stratum_sep	A character string to concatenate the columns in stratum, if multiple columns are provided.
alluvium	A character string of the column name to define the links. See data for more details.
alluvium_sep	A character string to concatenate the columns in alluvium, if multiple columns are provided.

split_by — The column(s) to split data by and plot separately.

split_by_sep — The separator for multiple split_by columns. See split_by

keep_empty — One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the x, group_by, fill_by, etc.

- FALSE (default): Drop empty factor levels from the data before plotting.
- TRUE: Keep empty factor levels and show them as a separate category in the plot.
- "level": Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: levels

flow — A logical value to use ggalluvial::geom_flow instead of ggalluvial::geom_alluvium.

expand — The values to expand the x and y axes. It is like CSS padding. When a single value is provided, it is used for both axes on both sides. When two values are provided, the first value is used for the top/bottom side and the second value is used for the left/right side. When three values are provided, the first value is used for the top side, the second value is used for the left/right side, and the third value is used for the bottom side. When four values are provided, the values are used for the top, right, bottom, and left sides, respectively. You can also use a named vector to specify the values for each side. When the axis is discrete, the values will be applied as 'add' to the 'expansion' function. When the axis is continuous, the values will be applied as 'mult' to the 'expansion' function. See also https://ggplot2.tidyverse.org/reference/expansion.html

nodes_legend — Controls how the legend of nodes will be shown. Possible values are:

- "merge": Merge the legends of nodes. That is only one legend will be shown for all nodes.
- "separate": Show the legends of nodes separately. That is, nodes on each x will have their own legend.
- "none": Do not show the legend of nodes.
- "auto": Automatically determine how to show the legend. When nodes_label is TRUE, "none" will apply. When nodes_label is FALSE, and if stratum is the same as links_fill_by, "none" will apply. If there is any overlapping values between the nodes on different x, "merge" will apply. Otherwise, "separate" will apply.

nodes_color — A character string to color the nodes. Use a special value ".fill" to use the same color as the fill.

links_fill_by — A character string of the column name to fill the links.

links_fill_by_sep — A character string to concatenate the columns in links_fill_by, if multiple columns are provided.

links_name — A character string to name the legend of links.

links_color — A character string to color the borders of links. Use a special value ".fill" to use the same color as the fill.

nodes_palette A character string to specify the palette of nodes fill.

nodes_palcolor A character vector to specify the colors of nodes fill.

nodes_alpha A numeric value to specify the transparency of nodes fill.

nodes_label A logical value to show the labels on the nodes.

nodes_label_miny A numeric value to specify the minimum y (frequency) to show the labels.

nodes_width A numeric value to specify the width of nodes.

links_palette A character string to specify the palette of links fill.

links_palcolor A character vector to specify the colors of links fill.

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

links_alpha A numeric value to specify the transparency of links fill.

legend.box A character string to specify the box of the legend, either "vertical" or "horizontal".

x_text_angle A numeric value specifying the angle of the x-axis text.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction A character string specifying the direction of the legend.

flip A logical value to flip the plot.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
axes	A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axes in individual plots. • 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout. • 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.
axis_titles	A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axis titles in individual plots. • 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction. • 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.
guides	A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'collect' will collect guides below to the given nesting level, removing duplicates. • 'keep' will stop collection at this level and let guides be placed alongside their plot. • 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.
design	Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
# Reproduce the examples in ggalluvial
set.seed(8525)

data(UCBAdmissions, package = "datasets")
UCBAdmissions <- as.data.frame(UCBAdmissions)
SankeyPlot(as.data.frame(UCBAdmissions), x = c("Gender", "Dept"),
```

```
    y = "Freq", nodes_width = 1/12, links_fill_by = "Admit", nodes_label = TRUE,
    nodes_palette = "simspec", links_palette = "Set1", links_alpha = 0.5,
    nodes_palcolor = "black", links_color = "transparent")

data(HairEyeColor, package = "datasets")
SankeyPlot(as.data.frame(HairEyeColor), x = c("Hair", "Eye", "Sex"),
    y = "Freq", links_fill_by = "Eye", nodes_width = 1/8, nodes_alpha = 0.4,
    flip = TRUE, reverse = FALSE, knot.pos = 0, links_color = "transparent",
    ylab = "Freq", links_alpha = 0.5, links_name = "Eye (links)", links_palcolor = c(
        Brown = "#70493D", Hazel = "#E2AC76", Green = "#3F752B", Blue = "#81B0E4"))

data(Refugees, package = "alluvial")
country_regions <- c(
    Afghanistan = "Middle East",
    Burundi = "Central Africa",
    `Congo DRC` = "Central Africa",
    Iraq = "Middle East",
    Myanmar = "Southeast Asia",
    Palestine = "Middle East",
    Somalia = "Horn of Africa",
    Sudan = "Central Africa",
    Syria = "Middle East",
    Vietnam = "Southeast Asia"
)
Refugees$region <- country_regions[Refugees$country]
SankeyPlot(Refugees, x = "year", y = "refugees", alluvium = "country",
    links_fill_by = "country", links_color = ".fill", links_alpha = 0.75,
   links_palette = "Set3", facet_by = "region", x_text_angle = -45, nodes_legend = "none",
    theme_args = list(strip.background = ggplot2::element_rect(fill="grey80")),
    decreasing = FALSE, nodes_width = 0, nodes_color = "transparent", ylab = "refugees",
    title = "Refugee volume by country and region of origin")

data(majors, package = "ggalluvial")
majors$curriculum <- as.factor(majors$curriculum)
SankeyPlot(majors, x = "semester", stratum = "curriculum", alluvium = "student",
    links_fill_by = "curriculum", flow = TRUE, stat = "alluvium", nodes_palette = "Set2",
    links_palette = "Set2")

data(vaccinations, package = "ggalluvial")
vaccinations <- transform(vaccinations,
    response = factor(response, rev(levels(response))))
SankeyPlot(vaccinations, x = "survey", stratum = "response", alluvium = "subject",
    y = "freq", links_fill_by = "response", nodes_label = TRUE, nodes_alpha = 0.5,
    nodes_palette = "seurat", links_palette = "seurat", links_alpha = 0.5,
    legend.position = "none", flow = TRUE, expand = c(0, 0, 0, .15), stat = "alluvium",
    title = "vaccination survey responses at three points in time")

data(Titanic, package = "datasets")
SankeyPlot(as.data.frame(Titanic), x = c("Class", "Sex"), y = "Freq",
    links_fill_by = "Survived", flow = TRUE, facet_by = "Age", facet_scales = "free_y",
    nodes_label = TRUE, expand = c(0.05, 0), xlab = "", links_palette = "Set1",
    nodes_palcolor = "white", nodes_label_miny = 10)
```

```
# Simulated examples
df <- data.frame(
    Clone = paste0("clone", 1:10),
    Timepoint1 = sample(c(rep(0, 30), 1:100), 10),
    Timepoint2 = sample(c(rep(0, 30), 1:100), 10)
)
SankeyPlot(df, x = c("Timepoint1", "Timepoint2"), alluvium = "Clone",
    links_color = ".fill")

df <- data.frame(
    Clone = rep(paste0("clone", 1:6), each = 2),
    Timepoint1 = sample(c(rep(0, 30), 1:100), 6),
    Timepoint2 = sample(c(rep(0, 30), 1:100), 6),
    Group = rep(c("A", "B"), 6)
)
SankeyPlot(df, x = c(".", "Timepoint1", "Timepoint2"),
    stratum = "Group", links_fill_by = "Clone", links_color = ".fill")
```

ScatterPlot *Scatter Plot*

Description

Scatter Plot

Usage

```
ScatterPlot(
  data,
  x,
  y,
  size_by = 2,
  size_name = NULL,
  color_by = NULL,
  color_name = NULL,
  palreverse = FALSE,
  split_by = NULL,
  split_by_sep = "_",
  shape = 21,
  alpha = ifelse(shape %in% 21:25, 0.65, 1),
  border_color = "black",
  highlight = NULL,
  highlight_shape = 16,
  highlight_size = 3,
  highlight_color = "red",
  highlight_alpha = 1,
  theme = "theme_this",
  theme_args = list(),
```

```
  palette = ifelse(!is.null(color_by) && !is.numeric(data[[color_by]]), "Paired",
    "Spectral"),
  palcolor = NULL,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
size_by	Which column to use as the size of the dots. It must be a numeric column. Or it can be a numeric value to specify the size of the dots.
size_name	A character vector specifying the name for the size legend.
color_by	Which column to use as the color of the dots. It could be a numeric column or a factor/character column. For shapes 21-25, the color is applied to the fill color.
color_name	A character vector specifying the name for the color legend.
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
shape	A numeric value specifying the shape of the points. Default is 21.
alpha	A numeric value specifying the transparency of the plot.

border_color A character vector specifying the color for the border of the points. Or TRUE to use the fill color as the border color.

highlight A vector of indexes or rownames to select the points to highlight. It could also be an expression (in string) to filter the data.

highlight_shape
A numeric value specifying the shape of the highlighted points. Default is 16.

highlight_size A numeric value specifying the size of the highlighted points. Default is 3.

highlight_color
A character vector specifying the color of the highlighted points. Default is "red".

highlight_alpha
A numeric value specifying the transparency of the highlighted points. Default is 1.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different `split_by` values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

facet_by A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by `split_by` and generate multiple plots and combine them into one using `patchwork::wrap_plots`

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See `ggplot2::facet_wrap`

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab	A character string specifying the y-axis label.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.
seed	The random seed to use. Default is 8525.
axes	A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axes in individual plots. • 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout. • 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.
axis_titles	A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'keep' will retain all axis titles in individual plots. • 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction. • 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.
guides	A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are: • 'collect' will collect guides below to the given nesting level, removing duplicates. • 'keep' will stop collection at this level and let guides be placed alongside their plot. • 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.
design	Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.
...	Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)

data <- data.frame(
   x = rnorm(20),
```

```
   y = rnorm(20),
   w = abs(rnorm(20)),
   t = sample(c("A", "B"), 20, replace = TRUE)
)
ScatterPlot(data, x = "x", y = "y")

# highlight points
ScatterPlot(data, x = "x", y = "y", highlight = 'x > 0')

# size_by is a numeric column
ScatterPlot(data, x = "x", y = "y", size_by = "w")

# color_by is a numeric column
ScatterPlot(data, x = "x", y = "y", color_by = "w")

# color_by is a factor/character column and set a border_color
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 border_color = "red")

# Same border_color as the fill color
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 border_color = TRUE)

# Shape doesn't have fill color
ScatterPlot(data, x = "x", y = "y", size_by = "w", color_by = "t",
 shape = 1, palette = "Set1")

# Change color per plot
ScatterPlot(data, x = "x", y = "y", split_by = "t",
            palcolor = list(A = "blue", B = "red"))
```

---

show_palettes *Show the color palettes*

Description

This function displays color palettes using ggplot2.

Usage

```
show_palettes(
  palettes = NULL,
  type = c("discrete", "continuous"),
  index = NULL,
  palette_names = NULL,
  return_names = TRUE,
  return_palettes = FALSE
)
```

Arguments

palettes	A list of color palettes. If NULL, uses default palettes.
type	A character vector specifying the type of palettes to include. Default is "discrete".
index	A numeric vector specifying the indices of the palettes to include. Default is NULL.
palette_names	A character vector specifying the names of the SCP palettes to include. Default is NULL.
return_names	A logical value indicating whether to return the names of the selected palettes. Default is TRUE.
return_palettes	A logical value indicating whether to return the colors of selected palettes. Default is FALSE.

Value

A list of palette names or a list of palettes.

See Also

palette_list

Examples

```
show_palettes(palettes = list(c("red", "blue", "green"), c("yellow", "purple", "orange")))
all_palettes <- show_palettes(return_palettes = TRUE)
names(all_palettes)
all_palettes[["simspec"]]
show_palettes(index = 1:10)
show_palettes(type = "discrete", index = 1:10)
show_palettes(type = "continuous", index = 1:10)
show_palettes(
    palette_names = c("Paired", "nejm", "simspec", "Spectral", "jet"),
    return_palettes = TRUE
)
```

SpatImagePlot *Plots for spatial elements*

Description

- SpatImagePlot: Plot a SpatRaster object as an image.
- SpatMasksPlot: Plot a SpatRaster object as masks.
- SpatShapesPlot: Plot a SpatVector object as shapes.
- SpatPointsPlot: Plot a data.frame of points with spatial coordinates.

Usage

```
SpatImagePlot(
  data,
  ext = NULL,
  raster = NULL,
  raster_dpi = NULL,
  flip_y = TRUE,
  palette = "turbo",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  return_layer = FALSE,
  theme = "theme_box",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

SpatMasksPlot(
  data,
  ext = NULL,
  flip_y = TRUE,
  add_border = TRUE,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  palette = "turbo",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  return_layer = FALSE,
  theme = "theme_box",
  theme_args = list(),
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)
```

```
SpatShapesPlot(
  data,
  x = NULL,
  y = NULL,
  group = NULL,
  ext = NULL,
  flip_y = TRUE,
  fill_by = NULL,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  palette = NULL,
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  return_layer = FALSE,
  theme = "theme_box",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

## S3 method for class 'SpatVector'
SpatShapesPlot(
  data,
  x = NULL,
  y = NULL,
  group = NULL,
  ext = NULL,
  flip_y = TRUE,
  fill_by = NULL,
  border_color = "black",
```

```
  border_size = 0.5,
  border_alpha = 1,
  palette = NULL,
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  return_layer = FALSE,
  theme = "theme_box",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

## S3 method for class 'data.frame'
SpatShapesPlot(
  data,
  x,
  y,
  group,
  ext = NULL,
  flip_y = TRUE,
  fill_by = "grey90",
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  palette = NULL,
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  fill_name = NULL,
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
```

```
  highlight_color = "black",
  highlight_stroke = 0.8,
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  return_layer = FALSE,
  theme = "theme_box",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)

SpatPointsPlot(
  data,
  x = NULL,
  y = NULL,
  ext = NULL,
  flip_y = TRUE,
  color_by = NULL,
  size_by = NULL,
  size = NULL,
  fill_by = NULL,
  lower_quantile = 0,
  upper_quantile = 0.99,
  lower_cutoff = NULL,
  upper_cutoff = NULL,
  palette = NULL,
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  color_name = NULL,
  size_name = NULL,
  shape = 16,
  border_color = "black",
  border_size = 0.5,
  border_alpha = 1,
  raster = NULL,
  raster_dpi = c(512, 512),
  hex = FALSE,
  hex_linewidth = 0.5,
  hex_count = FALSE,
  hex_bins = 50,
```

```
  hex_binwidth = NULL,
  label = FALSE,
  label_size = 4,
  label_fg = "white",
  label_bg = "black",
  label_bg_r = 0.1,
  label_repel = FALSE,
  label_repulsion = 20,
  label_pt_size = 1,
  label_pt_color = "black",
  label_segment_color = "black",
  label_insitu = FALSE,
 label_pos = c("median", "mean", "max", "min", "first", "last", "center", "random"),
  highlight = NULL,
  highlight_alpha = 1,
  highlight_size = 1,
  highlight_color = "black",
  highlight_stroke = 0.8,
  graph = NULL,
  graph_x = NULL,
  graph_y = NULL,
  graph_xend = NULL,
  graph_yend = NULL,
  graph_value = NULL,
  edge_size = c(0.05, 0.5),
  edge_alpha = 0.1,
  edge_color = "grey40",
  facet_scales = "fixed",
  facet_nrow = NULL,
  facet_ncol = NULL,
  facet_byrow = TRUE,
  return_layer = FALSE,
  theme = "theme_box",
  theme_args = list(),
  legend.position = ifelse(return_layer, "none", "right"),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  seed = 8525
)
```

Arguments

data	A SpatRaster or SpatVector object from the terra package, or a data.frame for SpatShapesPlot or SpatPointsPlot.
ext	A terra's SpatExtent object or a numeric vector of length 4 specifying the

	extent as `c(xmin, xmax, ymin, ymax)`. Default is NULL.
`raster`	Whether to raster the plot. Default is NULL.
`raster_dpi`	A numeric vector of the raster dpi. Default is c(512, 512).
`flip_y`	Whether to flip the y-axis direction. Default is TRUE. This is useful for visualizing spatial data with the origin at the top left corner.
`palette`	A character string specifying the color palette to use. For `SpatImagePlot`, if the data has 3 channels (RGB), it will be used as a color identity and this argument will be ignored.
`palcolor`	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different `split_by` values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
`palreverse`	A logical value indicating whether to reverse the palette. Default is FALSE.
`alpha`	A numeric value specifying the transparency of the plot.
`fill_name`	A character string for the fill legend title.
`return_layer`	Whether to return the layers or the plot. Default is FALSE.
`theme`	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
`theme_args`	A list of arguments to pass to the theme function.
`legend.position`	A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".
`legend.direction`	A character string specifying the direction of the legend.
`title`	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
`subtitle`	A character string specifying the subtitle of the plot.
`xlab`	A character string specifying the x-axis label.
`ylab`	A character string specifying the y-axis label.
`seed`	The random seed to use. Default is 8525.
`add_border`	Whether to add a border around the masks in `SpatMasksPlot`. Default is TRUE.
`border_color`	A character string of the border color. Default is "black".
`border_size`	A numeric value of the border width. Default is 0.5.
`border_alpha`	A numeric value of the border transparency. Default is 1.
`x`	A character string specifying the x-axis column name for `SpatPointsPlot` or `SpatShapesPlot` when `data` is a data.frame. If `data` is a `SpatRaster` or `SpatVector`, this argument is ignored.
`y`	A character string specifying the y-axis column name for `SpatPointsPlot` or `SpatShapesPlot` when `data` is a data.frame. If `data` is a `SpatRaster` or `SpatVector`, this argument is ignored.

group A character string specifying the grouping column for SpatShapesPlot when data is a data.frame.

fill_by A character string or vector specifying the column(s) to fill the shapes in SpatShapesPlot.

highlight A character vector of the row names to highlight. Default is NULL.

highlight_alpha
A numeric value of the highlight transparency. Default is 1.

highlight_size A numeric value of the highlight size. Default is 1.

highlight_color
A character string of the highlight color. Default is "black".

highlight_stroke
A numeric value of the highlight stroke. Default is 0.8.

facet_scales Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap

facet_nrow A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_ncol A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow A logical value indicating whether to fill the plots by row. Default is TRUE.

color_by A character string specifying the column to color the points in SpatPointsPlot.

size_by A character string specifying the column to size the points in SpatPointsPlot.

size Alias of size_by when size is a numeric value.

lower_quantile, upper_quantile, lower_cutoff, upper_cutoff
Vector of minimum and maximum cutoff values or quantile values for each numeric value.

color_name A character string for the color legend title in SpatPointsPlot.

size_name A character string for the size legend title in SpatPointsPlot.

shape A numeric value or character string specifying the shape of the points in SpatPointsPlot.

hex Whether to use hex plot. Default is FALSE.

hex_linewidth A numeric value of the hex line width. Default is 0.5.

hex_count Whether to count the hex.

hex_bins A numeric value of the hex bins. Default is 50.

hex_binwidth A numeric value of the hex bin width. Default is NULL.

label Whether to show the labels of groups. Default is FALSE.

label_size A numeric value of the label size. Default is 4.

label_fg A character string of the label foreground color. Default is "white".

label_bg A character string of the label background color. Default is "black".

label_bg_r A numeric value of the background ratio of the labels. Default is 0.1.

label_repel Whether to repel the labels. Default is FALSE.

label_repulsion
A numeric value of the label repulsion. Default is 20.

label_pt_size A numeric value of the label point size. Default is 1.

label_pt_color A character string of the label point color. Default is "black".

label_segment_color
A character string of the label segment color. Default is "black".

label_insitu Whether to place the raw labels (group names) in the center of the points with the corresponding group. Default is FALSE, which uses numbers instead of raw labels.

label_pos A character string or a function specifying the position of the labels.

- "mean": Place labels at the mean position of the points in each group. Same as `function(x) mean(x, na.rm = TRUE)`.
- "center": Place labels at the center of the points in each group. Same as `function(x) mean(range(x, na.rm = TRUE))`.
- "median": Place labels at the median position of the points in each group. Same as `function(x) median(x, na.rm = TRUE)`.
- "first": Place labels at the first point in each group. Same as `function(x) x[1]`.
- "last": Place labels at the last point in each group. Same as `function(x) x[length(x)]`.
- "random": Place labels at a random point in each group. Same as `function(x) sample(x, 1)`.
- "min": Place labels at the minimum position (both x and y) of the points in each group. Same as `function(x) min(x, na.rm = TRUE)`.
- "max": Place labels at the maximum position (both x and y) of the points in each group. Same as `function(x) max(x, na.rm = TRUE)`.

graph A character string of column names or the indexes in the data for the graph data. Default is NULL. If "@graph" is provided, the graph data will be extracted from the data attribute 'graph'. The graph data should be an adjacency matrix (numeric matrix) with row and column names matching the point IDs. Or a data.frame with x, xend, y, yend and value columns. If so, `graph_x`, `graph_y`, `graph_xend`, `graph_yend`, and `graph_value` arguments can be used to specify the column names.

graph_x A character string of the x column name for the graph data.

graph_y A character string of the y column name for the graph data.

graph_xend A character string of the xend column name for the graph data.

graph_yend A character string of the yend column name for the graph data.

graph_value A character string of the value column name for the graph data.

edge_size A numeric vector of the edge size range. Default is c(0.05, 0.5).

edge_alpha A numeric value of the edge transparency. Default is 0.1.

edge_color A character string of the edge color. Default is "grey40".

Examples

```
set.seed(8525)
# --- SpatImagePlot ---
# Generate a sample SpatRaster
r <- terra::rast(
    nrows = 50, ncols = 40, vals = runif(2000),
    xmin = 0, xmax = 40, ymin = 0, ymax = 50,
    crs = ""
)
SpatImagePlot(r)
SpatImagePlot(r, raster = TRUE, raster_dpi = 20)
SpatImagePlot(r, alpha = 0.5, theme = "theme_blank",
    theme_args = list(add_coord = FALSE), fill_name = "value")
SpatImagePlot(r, ext = c(0, 10, 0, 10), flip_y = FALSE, palette = "viridis")

# --- SpatMasksPlot ---
m <- terra::rast(
   nrows = 50, ncols = 40,
   vals = sample(c(1:5, NA), 2000, replace = TRUE, prob = c(rep(0.04, 5), 0.8)),
   xmin = 0, xmax = 40, ymin = 0, ymax = 50,
   crs = ""
)
SpatMasksPlot(m, border_color = "red")
SpatMasksPlot(m, ext = c(0, 15, 0, 20), add_border = FALSE,
    palreverse = TRUE, fill_name = "value")

# --- SpatShapesPlot ---
polygons <- data.frame(
   id = paste0("poly_", 1:10),
   cat = sample(LETTERS[1:3], 10, replace = TRUE),
   feat1 = rnorm(10),
   feat2 = rnorm(10),
   geometry = c(
       'POLYGON((64.6 75.3,66.0 70.5,66.4 70.2,67.0 69.8,72.8 70.4,64.6 75.3))',
       'POLYGON((56.7 63.0,52.3 65.6,48.0 63.2,51.2 55.7,57.1 59.2,56.7 63.0))',
       'POLYGON((9.9 16.5,9.3 15.9,8.0 13.1,11.5 7.8,17.8 11.3,9.9 16.5))',
       'POLYGON((64.9 37.2,60.3 37.4,57.6 31.7,58.9 29.3,64.0 28.1,64.9 37.2))',
       'POLYGON((30.5 49.1,22.4 46.5,22.4 43.9,30.9 41.9,31.6 42.9,30.5 49.1))',
       'POLYGON((78.3 57.8,70.5 61.6,71.6 52.7,72.2 52.5,77.4 54.5,78.3 57.8))',
       'POLYGON((41.8 23.8,41.3 25.9,41.0 26.4,36.5 28.7,35.8 28.6,41.8 23.8))',
       'POLYGON((15.7 75.9,14.2 74.4,15.7 67.5,23.0 69.8,23.4 71.7,15.7 75.9))',
       'POLYGON((80.7 37.4,75.3 31.3,77.1 28.5,82.5 28.0,83.1 28.5,80.7 37.4))',
       'POLYGON((15.5 37.8,14.4 38.6,7.3 32.6,8.3 30.9,15.1 30.2,15.5 37.8))'
   )
)

polygons <- terra::vect(polygons, crs = "EPSG:4326", geom = "geometry")

SpatShapesPlot(polygons)
SpatShapesPlot(polygons, ext = c(0, 20, 0, 20))
SpatShapesPlot(polygons, highlight = 'cat == "A"', highlight_color = "red2")
SpatShapesPlot(polygons, border_color = "red", border_size = 2)
```

```
SpatShapesPlot(polygons, fill_by = "cat", fill_name = "category")
# Let border color be determined by fill
SpatShapesPlot(polygons, fill_by = "cat", alpha = 0.6, border_color = TRUE)
SpatShapesPlot(polygons, fill_by = "feat1")
SpatShapesPlot(polygons, fill_by = c("feat1", "feat2"), palette = "RdYlBu")

# --- SpatPointsPlot ---
# create some random points in the above polygons
points <- data.frame(
  id = paste0("point_", 1:30),
  gene = sample(LETTERS[1:3], 30, replace = TRUE),
  feat1 = runif(30, 0, 100),
  feat2 = runif(30, 0, 100),
  size = runif(30, 1, 5),
  x = c(
    61.6, 14.3, 12.7, 49.6, 74.9, 58.9, 13.9, 24.7, 16.9, 15.6,
    72.4, 60.1, 75.4, 14.9, 80.3, 78.8, 16.7, 27.6, 48.9, 52.5,
    12.9, 11.8, 50.4, 25.6, 10.4, 51.9, 73.4, 26.8, 50.4, 60.0
  ),
  y = c(
    32.1, 12.8, 33.2, 59.9, 57.8, 31.9, 10.1, 46.8, 75.3, 69.0,
    60.0, 29.4, 54.2, 34.2, 35.3, 33.1, 74.7, 48.0, 63.2, 59.2,
    9.2, 15.1, 64.5, 47.1, 11.4, 60.1, 54.1, 44.5, 61.9, 30.3
  )
)

SpatPointsPlot(points)
SpatPointsPlot(points, color_by = "gene", size_by = "size", shape = 22,
  border_size = 1)
SpatPointsPlot(points, raster = TRUE, raster_dpi = 30, color_by = "feat1")
SpatPointsPlot(points, color_by = c("feat1", "feat2"), size_by = "size")
SpatPointsPlot(points, color_by = "feat1", upper_cutoff = 50)
SpatPointsPlot(points, color_by = "feat1", hex = TRUE)
SpatPointsPlot(points, color_by = "gene", label = TRUE)
SpatPointsPlot(points, color_by = "gene", highlight = 1:20,
  highlight_color = "red2", highlight_stroke = 0.8)

# --- Graph/Network functionality ---
# Create a simple adjacency matrix for demonstration
set.seed(8525)
graph_mat <- matrix(0, nrow = 30, ncol = 30)
# Add some random connections with weights
for(i in 1:30) {
  neighbors <- sample(setdiff(1:30, i), size = sample(2:5, 1))
  graph_mat[i, neighbors] <- runif(length(neighbors), 0.1, 1)
}
rownames(graph_mat) <- colnames(graph_mat) <- rownames(points)
attr(points, "graph") <- graph_mat

SpatPointsPlot(points, color_by = "gene", graph = "@graph",
  edge_color = "grey60", edge_alpha = 0.3)
SpatPointsPlot(points, color_by = "feat1", graph = graph_mat,
  edge_size = c(0.1, 1), edge_alpha = 0.5)
```

```
# --- Use the `return_layer` argument to get the ggplot layers
ext = c(0, 40, 0, 50)
ggplot2::ggplot() +
  SpatImagePlot(r, return_layer = TRUE, alpha = 0.2, ext = ext) +
  SpatShapesPlot(polygons, return_layer = TRUE, ext = ext, fill_by = "white") +
  SpatPointsPlot(points, return_layer = TRUE, ext = ext, color_by = "feat1") +
  theme_box() +
  ggplot2::coord_sf(expand = 0) +
  ggplot2::scale_y_continuous(labels = function(x) -x)
```

`theme_blank` *Blank theme*

Description

This function creates a theme with all elements blank except for axis lines and labels. It can optionally add coordinate axes in the plot.

Usage

```
theme_blank(
  add_coord = TRUE,
  xlen_npc = 0.15,
  ylen_npc = 0.15,
  xlab = "",
  ylab = "",
  lab_size = 12,
  ...
)
```

Arguments

`add_coord`	Whether to add coordinate arrows. Default is `TRUE`.
`xlen_npc`	The length of the x-axis arrow in "npc".
`ylen_npc`	The length of the y-axis arrow in "npc".
`xlab`	x-axis label.
`ylab`	y-axis label.
`lab_size`	Label size.
`...`	Arguments passed to the `theme`.

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(cyl))) +
    geom_point()
p + theme_blank()
p + theme_blank(xlab = "x-axis", ylab = "y-axis", lab_size = 16)
```

`theme_box` *Box theme*

Description

This function creates a theme with all elements blank except for axis lines like a box around the plot.

Usage

```
theme_box(
  xlen_npc = 0.15,
  ylen_npc = 0.15,
  xlab = "",
  ylab = "",
  lab_size = 12,
  ...
)
```

Arguments

`xlen_npc`	The length of the x-axis arrow in "npc".
`ylen_npc`	The length of the y-axis arrow in "npc".
`xlab`	x-axis label.
`ylab`	y-axis label.
`lab_size`	Label size.
`...`	Arguments passed to the `theme`.

Value

A ggplot2 theme.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(cyl))) +
    geom_point()
p + theme_box()
```

theme_this *A ggplot2 theme and palettes for plotthis Borrowed from the* theme_this *function in the* SCP *pipeline*

Description

A ggplot2 theme and palettes for plotthis Borrowed from the theme_this function in the SCP pipeline

Usage

```
theme_this(aspect.ratio = NULL, base_size = NULL, font_family = NULL, ...)
```

Arguments

aspect.ratio	The aspect ratio of the plot
base_size	The base size of the text If not specified, it will use the value from getOption("theme_this.base_size" 12). If you want to change the default base size, you can set the option theme_this.base_size. This is applied to all plots using this theme.
font_family	The font family of the text If not specified, it will use the value from getOption("theme_this.font_fam If you want to change the default font family, you can set the option theme_this.font_family. This is applied to all plots using this theme. To list available font families, you can use the systemfonts::system_fonts() function.
...	Other arguments for theme()

Value

A ggplot2 theme

See Also

https://github.com/zhanghao-njmu/SCP

TrendPlot *Trend plot*

Description

A trend plot is like an area plot but with gaps between the bars.

Usage

```
TrendPlot(
  data,
  x,
  y = NULL,
  x_sep = "_",
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  group_name = NULL,
  scale_y = FALSE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Paired",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  x_text_angle = 0,
  aspect.ratio = 1,
  legend.position = waiver(),
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  keep_na = FALSE,
  keep_empty = FALSE,
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.

x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
x_sep	A character string to concatenate the columns in x, if multiple columns are provided.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
group_name	A character string to name the legend of fill.
scale_y	A logical value to scale the y-axis by the total number in each x-axis group.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
alpha	A numeric value specifying the transparency of the plot.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
x_text_angle	A numeric value specifying the angle of the x-axis text.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

xlab — A character string specifying the x-axis label.

ylab — A character string specifying the y-axis label.

keep_na — A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

keep_empty — One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

- `FALSE` (default): Drop empty factor levels from the data before plotting.
- `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
- `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

seed — The random seed to use. Default is 8525.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.

- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

See Also

`AreaPlot`

Examples

```
data <- data.frame(
    x = factor(rep(c("A", "B", NA, "D"), 3), levels = LETTERS[1:4]),
    y = c(1, 3, 6, 5, 4, 2, 5, 7, 8, 9, 4, 8),
    group = factor(rep(c("F1", NA, "F3"), each = 4), levels = c("F1", "F2", "F3"))
)
TrendPlot(data, x = "x", y = "y", group_by = "group")
TrendPlot(data, x = "x", y = "y", group_by = "group",
         scale_y = TRUE)
TrendPlot(data, x = "x", y = "y", split_by = "group")
TrendPlot(data, x = "x", y = "y", split_by = "group",
          palette = c(F1 = "Set1", F3 = "Dark2"))
TrendPlot(data, x = "x", y = "y", group_by = "group",
         keep_na = TRUE, keep_empty = TRUE)
TrendPlot(data, x = "x", y = "y", group_by = "group",
         keep_na = TRUE, keep_empty = list(x = FALSE, group = 'level'))
TrendPlot(data, x = "x", y = "y", facet_by = "group",
         keep_na = TRUE, keep_empty = list(x = FALSE, group = 'level'))
```

UpsetPlot *Upset Plot*

Description

Upset Plot

Usage

```
UpsetPlot(
  data,
  in_form = c("auto", "long", "wide", "list", "upset"),
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  id_by = NULL,
  label = TRUE,
  label_fg = "black",
  label_size = NULL,
  label_bg = "white",
  label_bg_r = 0.1,
  palette = "Blues",
  palcolor = NULL,
  palreverse = FALSE,
  alpha = 1,
  specific = TRUE,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 0.6,
  legend.position = "right",
  legend.direction = "vertical",
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  combmatrix_gap = 6,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
```

```
)
```

Arguments

data	A data frame.

in_form	A character string indicating the datatype of the input data. Possible values are "long", "wide", "list", "upset" or "auto". "long" indicates the data is in long format. "wide" indicates the data is in wide format. "list" indicates the data is a list. "upset" indicates the data is a UpsetPlotData object. "auto" indicates the function will detect the datatype of the input data.
	A long format data would look like:
	`group_by id_by`
	`A a1`
	`A a2`
	`B a1`
	`B a3`
	`...`
	A wide format data would look like:
	`A B`
	`TRUE TRUE`
	`TRUE FALSE`
	`FALSE TRUE`
	`...`
	A list format data would look like:
	`list(A = c("a1", "a2"), B = c("a1", "a3"))`
	An UpsetPlotData object is generated by prepare_update_data() would look like:
	`group_by`
	`--------`
	`list("A")  # a2`
	`list("B")  # a3`
	`list(c("A", "B"))  # a1`
	`...`

split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
id_by	A character string specifying the column name of the data frame to identify the instances. Required when group_by is a single column and data is a data frame.
label	A logical value to show the labels on the bars.
label_fg	A character string specifying the color of the label text.

label_size A numeric value specifying the size of the label text.

label_bg A character string specifying the background color of the label.

label_bg_r A numeric value specifying the radius of the background of the label.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

alpha A numeric value specifying the transparency of the plot.

specific A logical value to show the specific intersections only. ggVennDiagram, by default, only return the specific subsets of a region. However, sometimes, we want to show all the overlapping items for two or more sets. See https://github.com/gaospecial/ggVennDiagram/issues/64 for more details.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

xlab A character string specifying the x-axis label.

ylab A character string specifying the y-axis label.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

legend.position
A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction
A character string specifying the direction of the legend.

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow A logical value indicating whether to fill the plots by row.

seed The random seed to use. Default is 8525.

combmatrix_gap A numeric value specifying the gap between the rows of the combination matrix. The default value is 6, which is suitable for a base_size of 12. The actual gap will be scaled by the text size scale, which is calculated as base_size / 12.

axes A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.

- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

`axis_titles` A string specifying how axis titltes should be treated. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

`guides` A string specifying how guides should be treated in the layout. Passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

`design` Specification of the location of areas in the layout, passed to `patchwork::wrap_plots()`. Only relevant when `split_by` is used and `combine` is TRUE. When specified, `nrow`, `ncol`, and `byrow` are ignored. See `patchwork::wrap_plots()` for more details.

`...` Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- list(
    A = 1:5,
    B = 2:6,
    C = 3:7,
    D = 4:8
)
UpsetPlot(data)
UpsetPlot(data, label = FALSE)
UpsetPlot(data, palette = "Reds", specific = FALSE)

# long form input
data_long <- data.frame(
    group_by = factor(
         c(rep("A", 5), rep("B", 5), rep("C", 5), rep("D", 5)),
         levels = c("A", "B", "C", "D")
    ),
    id_by = c(1:5, 2:6, 3:7, 4:8)
```

```
)
UpsetPlot(data_long, in_form = "long", group_by = "group_by", id_by = "id_by")

# wide form input
data <- data.frame(
    id = LETTERS[1:10],
    B = c(1, 0, 1, 1, 0, 0, 1, 0, 1, 0),
    A = c(1, 1, 1, 0, 0, 1, 0, 0, 1, 0),
    D = c(1, 0, 0, 1, 1, 0, 0, 1, 0, 1),
    C = c(0, 1, 1, 0, 1, 0, 1, 0, 1, 0)
)
UpsetPlot(data, in_form = "wide", id_by = "id", n_intersections = 4)
```

`VelocityPlot` *Cell velocity plot*

Description

The plot shows the velocity vectors of the cells in a specified reduction space.

Usage

```
VelocityPlot(
  embedding,
  v_embedding,
  plot_type = c("raw", "grid", "stream"),
  split_by = NULL,
  group_by = NULL,
  group_name = "Group",
  group_palette = "Paired",
  group_palcolor = NULL,
  n_neighbors = NULL,
  density = 1,
  smooth = 0.5,
  scale = 1,
  min_mass = 1,
  cutoff_perc = 5,
  arrow_angle = 20,
  arrow_color = "black",
  arrow_alpha = 1,
  keep_na = FALSE,
  keep_empty = FALSE,
  streamline_l = 5,
  streamline_minl = 1,
  streamline_res = 1,
  streamline_n = 15,
  streamline_width = c(0, 0.8),
```

```
  streamline_alpha = 1,
  streamline_color = NULL,
  streamline_palette = "RdYlBu",
  streamline_palcolor = NULL,
  palreverse = FALSE,
  streamline_bg_color = "white",
  streamline_bg_stroke = 0.5,
  aspect.ratio = 1,
  title = "Cell velocity",
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  theme = "theme_this",
  theme_args = list(),
  return_layer = FALSE,
  seed = 8525
)
```

Arguments

embedding	A matrix or data.frame of dimension n_obs x n_dim specifying the embedding coordinates of the cells.
v_embedding	A matrix or data.frame of dimension n_obs x n_dim specifying the velocity vectors of the cells.
plot_type	A character string specifying the type of plot to create. Options are "raw", "grid", or "stream". Default is "raw".
split_by	An optional character string specifying a variable to split the plot by. Not supported yet.
group_by	An optional character string specifying a variable to group the cells by.
group_name	An optional character string specifying the name of the grouping variable in legend. Default is "Group".
group_palette	A character string specifying the color palette to use for grouping. Default is "Paired".
group_palcolor	An optional character vector specifying the colors to use for grouping. If NULL, the colors will be generated from the group_palette.
n_neighbors	An optional numeric value specifying the number of nearest neighbors for each grid point. Default is ceiling(ncol(embedding) / 50).
density	An optional numeric value specifying the density of the grid points along each dimension. Default is 1.
smooth	An optional numeric value specifying the smoothing factor for the velocity vectors. Default is 0.5.
scale	An optional numeric value specifying the scaling factor for the velocity vectors. Default is 1.

`min_mass` An optional numeric value specifying the minimum mass required for a grid point to be considered. Default is 1.

`cutoff_perc` An optional numeric value specifying the percentile cutoff for removing low-density grid points. Default is 5.

`arrow_angle` An optional numeric value specifying the angle of the arrowheads in degrees for velocity arrows. Default is 20.

`arrow_color` A character string specifying the color of the velocity arrowheads. Default is "black".

`arrow_alpha` A numeric value specifying the transparency of the velocity arrows. Default is 1 (fully opaque). Only works for `plot_type = "raw"` and `plot_type = "grid"`. For `plot_type = "stream"`, use `streamline_alpha` instead.

`keep_na` A logical value or a character to replace the NA values in the data. It can also take a named list to specify different behavior for different columns. If TRUE or NA, NA values will be replaced with NA. If FALSE, NA values will be removed from the data before plotting. If a character string is provided, NA values will be replaced with the provided string. If a named vector/list is provided, the names should be the column names to apply the behavior to, and the values should be one of TRUE, FALSE, or a character string. Without a named vector/list, the behavior applies to categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

`keep_empty` One of FALSE, TRUE and "level". It can also take a named list to specify different behavior for different columns. Without a named list, the behavior applies to the categorical/character columns used on the plot, for example, the `x`, `group_by`, `fill_by`, etc.

- `FALSE` (default): Drop empty factor levels from the data before plotting.
- `TRUE`: Keep empty factor levels and show them as a separate category in the plot.
- `"level"`: Keep empty factor levels, but do not show them in the plot. But they will be assigned colors from the palette to maintain consistency across multiple plots. Alias: `levels`

`streamline_l` An optional numeric value specifying the length of the velocity streamlines. Default is 5.

`streamline_minl` An optional numeric value specifying the minimum length of the velocity streamlines. Default is 1.

`streamline_res` An optional numeric value specifying the resolution of the velocity streamlines. Default is 1.

`streamline_n` An optional numeric value specifying the number of velocity streamlines to draw. Default is 15.

`streamline_width` A numeric vector of length 2 specifying the width of the velocity streamlines. Default is c(0, 0.8).

`streamline_alpha` A numeric value specifying the transparency of the velocity streamlines. Default is 1 (fully opaque).

streamline_color
: A character string specifying the color of the velocity streamlines.

streamline_palette
: A character string specifying the color palette to use for the velocity streamlines. Default is "RdYlBu".

streamline_palcolor
: An optional character vector specifying the colors to use for the velocity streamlines. If NULL, the colors will be generated from the streamline_palette.

palreverse
: A logical value indicating whether to reverse the palette. Default is FALSE.

streamline_bg_color
: A character string specifying the background color of the velocity streamlines. Default is "white".

streamline_bg_stroke
: A numeric value specifying the background stroke width of the velocity streamlines. Default is 0.5.

aspect.ratio
: A numeric value specifying the aspect ratio of the plot.

title
: A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle
: A character string specifying the subtitle of the plot.

xlab
: A character string specifying the x-axis label.

ylab
: A character string specifying the y-axis label.

legend.position
: A character string specifying the position of the legend. if `waiver()`, for single groups, the legend will be "none", otherwise "right".

legend.direction
: A character string specifying the direction of the legend.

theme
: A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args
: A list of arguments to pass to the theme function.

return_layer
: A logical value indicating whether to return the ggplot layer instead of the full plot. Default is FALSE.

seed
: The random seed to use. Default is 8525.

Value

A ggplot object representing the cell velocity plot or a ggplot layer if `return_layer` is TRUE.

See Also

`DimPlot()` `FeatureDimPlot()`

Examples

```
data(dim_example)
dim_example$clusters[dim_example$clusters == "Ductal"] <- NA

VelocityPlot(dim_example[, 1:2], dim_example[, 3:4], group_by = dim_example$clusters)
VelocityPlot(dim_example[, 1:2], dim_example[, 3:4], group_by = dim_example$clusters,
    keep_na = TRUE, keep_empty = TRUE)
VelocityPlot(dim_example[, 1:2], dim_example[, 3:4], group_by = dim_example$clusters,
    keep_na = TRUE, keep_empty = 'level')
VelocityPlot(dim_example[, 1:2], dim_example[, 3:4], group_by = dim_example$clusters,
    keep_na = TRUE, keep_empty = FALSE)
```

VennDiagram *Venn diagram*

Description

Venn diagram

Usage

```
VennDiagram(
  data,
  in_form = c("auto", "long", "wide", "list", "venn"),
  split_by = NULL,
  split_by_sep = "_",
  group_by = NULL,
  group_by_sep = "_",
  id_by = NULL,
  label = "count",
  label_fg = "black",
  label_size = NULL,
  label_bg = "white",
  label_bg_r = 0.1,
  fill_mode = "count",
  palreverse = FALSE,
  fill_name = NULL,
  palette = ifelse(fill_mode == "set", "Paired", "Blues"),
  palcolor = NULL,
  alpha = 1,
  theme = "theme_this",
  theme_args = list(),
  title = NULL,
  subtitle = NULL,
  legend.position = "right",
  legend.direction = "vertical",
```

```
  aspect.ratio = 1,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  seed = 8525,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.

in_form	A character string indicating the datatype of the input data. Possible values are "long", "wide", "list", "venn" or NULL. "long" indicates the data is in long format. "wide" indicates the data is in wide format. "list" indicates the data is a list. "venn" indicates the data is a VennPlotData object. "auto" indicates the function will detect the datatype of the input data.
	A long format data would look like:
	`group_by id_by`
	`A a1`
	`A a2`
	`B a1`
	`B a3`
	`...`
	A wide format data would look like:
	`A B`
	`TRUE TRUE`
	`TRUE FALSE`
	`FALSE TRUE`
	`...`
	A list format data would look like:
	`list(A = c("a1", "a2"), B = c("a1", "a3"))`

split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
group_by	Columns to group the data for plotting For those plotting functions that do not support multiple groups, They will be concatenated into one column, using group_by_sep as the separator
group_by_sep	The separator for multiple group_by columns. See group_by
id_by	A character string specifying the column name of the data frame to identify the instances. Required when group_by is a single column and data is a data frame.

label A character string specifying the label to show on the Venn diagram. Possible values are "count", "percent", "both", "none" and a function. "count" indicates the count of the intersection. "percent" indicates the percentage of the intersection. "both" indicates both the count and the percentage of the intersection. "none" indicates no label. If it is a function, if takes a data frame as input and returns a character vector as label. The data frame has columns "id", "X", "Y", "name", "item" and "count".

label_fg A character string specifying the color of the label text.

label_size A numeric value specifying the size of the label text.

label_bg A character string specifying the background color of the label.

label_bg_r A numeric value specifying the radius of the background of the label.

fill_mode A character string specifying the fill mode of the Venn diagram. Possible values are "count", "set", "count_rev". "count" indicates the fill color is based on the count of the intersection. "set" indicates the fill color is based on the set of the intersection. "count_rev" indicates the fill color is based on the count of the intersection in reverse order. The palette will be continuous for "count" and "count_rev". The palette will be discrete for "set".

palreverse A logical value indicating whether to reverse the palette. Default is FALSE.

fill_name A character string to name the legend of colorbar.

palette A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha A numeric value specifying the transparency of the plot.

theme A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args A list of arguments to pass to the theme function.

title A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle A character string specifying the subtitle of the plot.

legend.position A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction A character string specifying the direction of the legend.

aspect.ratio A numeric value specifying the aspect ratio of the plot.

combine Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow A numeric value specifying the number of rows in the facet.

ncol A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

seed — The random seed to use. Default is 8525.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A combined ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
set.seed(8525)
data = list(
    A = sort(sample(letters, 8)),
    B = sort(sample(letters, 8)),
    C = sort(sample(letters, 8)),
    D = sort(sample(letters, 8))
)

VennDiagram(data)
VennDiagram(data, fill_mode = "set")
```

```
VennDiagram(data, label = "both")
# label with a function
VennDiagram(data, label = function(df) df$name)
VennDiagram(data, palette = "material-indigo", alpha = 0.6)
```

VolcanoPlot *Volcano plot*

Description

A volcano plot is a type of scatter plot that shows statistical significance (usually on the y-axis) versus magnitude of change (usually on the x-axis).

Usage

```
VolcanoPlot(
  data,
  x,
  y,
  ytrans = function(n) -log10(n),
  color_by = NULL,
  color_name = NULL,
  xlim = NULL,
  flip_negatives = FALSE,
  x_cutoff = NULL,
  y_cutoff = 0.05,
  split_by = NULL,
  split_by_sep = "_",
  label_by = NULL,
  x_cutoff_name = NULL,
  y_cutoff_name = NULL,
  x_cutoff_color = "red2",
  y_cutoff_color = "blue2",
  x_cutoff_linetype = "dashed",
  y_cutoff_linetype = "dashed",
  x_cutoff_linewidth = 0.5,
  y_cutoff_linewidth = 0.5,
  pt_size = 2,
  pt_alpha = 0.5,
  nlabel = 5,
  labels = NULL,
  label_size = 3,
  label_fg = "black",
  label_bg = "white",
  label_bg_r = 0.1,
  highlight = NULL,
```

```
  highlight_color = "red",
  highlight_size = 2,
  highlight_alpha = 1,
  highlight_stroke = 0.5,
  trim = c(0, 1),
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  palreverse = FALSE,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
x	A character string specifying the column name of the data frame to plot for the x-axis.
y	A character string specifying the column name of the data frame to plot for the y-axis.
ytrans	A function to transform the y-axis values.
color_by	A character vector of column names to color the points by. If NULL, the points will be filled by the x and y cutoff value.
color_name	A character string to name the legend of color.
xlim	A numeric vector of length 2 to set the x-axis limits.
flip_negatives	A logical value to flip the y-axis for negative x values.

x_cutoff A numeric value to set the x-axis cutoff. Both negative and positive of this value will be used.

y_cutoff A numeric value to set the y-axis cutoff. Note that the y-axis cutoff will be transformed by ytrans. So you should provide the original value.

split_by The column(s) to split data by and plot separately.

split_by_sep The separator for multiple split_by columns. See split_by

label_by A character string of column name to use as labels. If NULL, the row names will be used.

x_cutoff_name A character string to name the x-axis cutoff. If "none", the legend for the x-axis cutoff will not be shown.

y_cutoff_name A character string to name the y-axis cutoff. If "none", the legend for the y-axis cutoff will not be shown.

x_cutoff_color A character string to color the x-axis cutoff line.

y_cutoff_color A character string to color the y-axis cutoff line.

x_cutoff_linetype
A character string to set the x-axis cutoff line type.

y_cutoff_linetype
A character string to set the y-axis cutoff line type.

x_cutoff_linewidth
A numeric value to set the x-axis cutoff line size.

y_cutoff_linewidth
A numeric value to set the y-axis cutoff line size.

pt_size A numeric value to set the point size.

pt_alpha A numeric value to set the point transparency.

nlabel A numeric value to set the number of labels to show. The points will be ordered by the distance to the origin. Top nlabel points will be labeled.

labels A character vector of row names or indexes to label the points.

label_size A numeric value to set the label size.

label_fg A character string to set the label color.

label_bg A character string to set the label background color.

label_bg_r A numeric value specifying the radius of the background of the label.

highlight A character vector of row names or indexes to highlight the points.

highlight_color
A character string to set the highlight color.

highlight_size A numeric value to set the highlight size.

highlight_alpha
A numeric value to set the highlight transparency.

highlight_stroke
A numeric value to set the highlight stroke size.

trim	A numeric vector of length 2 to trim the x-axis values. The values must be in the range from 0 to 1, which works as quantile to trim the x-axis values. For example, c(0.01, 0.99) will trim the 1% and 99% quantile of the x-axis values. If the values are less then 1% or greater than 99% quantile, the values will be set to the 1% or 99% quantile.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.
facet_byrow	A logical value indicating whether to fill the plots by row. Default is TRUE.
theme	A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".
theme_args	A list of arguments to pass to the theme function.
palette	A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.
palcolor	A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).
palreverse	A logical value indicating whether to reverse the palette. Default is FALSE.
title	A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.
subtitle	A character string specifying the subtitle of the plot.
xlab	A character string specifying the x-axis label.
ylab	A character string specifying the y-axis label.
aspect.ratio	A numeric value specifying the aspect ratio of the plot.
legend.position	A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".
legend.direction	A character string specifying the direction of the legend.
seed	The random seed to use. Default is 8525.
combine	Whether to combine the plots into one when facet is FALSE. Default is TRUE.
nrow	A numeric value specifying the number of rows in the facet.
ncol	A numeric value specifying the number of columns in the facet.
byrow	A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A list of ggplot objects or a wrap_plots object

Examples

```
set.seed(8525)
# Obtained by Seurat::FindMakers for the first cluster of pbmc_small
data <- data.frame(
   avg_log2FC = c(
     -3.69, -4.10, -2.68, -3.51, -3.09, -2.52, -3.53, -3.35, -2.82, -2.71, -3.16, -2.24,
     -5.62, -3.10, -3.42, -2.72, -3.23, -3.25, -4.68, 3.67, -2.66, 4.79, -2.99, 10.14,
     -1.78, -2.67, -2.26, -2.59, -3.39, 5.36, 4.56, 4.62, -2.94, -9.47, -9.12, -1.63,
     -2.77, 3.31, -1.53, -3.89, -4.21, 4.72, -2.98, -2.29, -1.41, -9.48, -4.30, 3.01,
     -1.19, -4.83, -1.35, -1.68, -1.63, -2.70, 3.86, 3.81, 7.23, -1.45, -0.92, -2.45,
     3.91, -4.45, -9.33, 3.56, 2.27, -1.60, -1.15, 11.40, -9.77, -8.32, 2.61, -1.25,
     -1.72, 10.61, 11.34, 10.02, 2.78, -3.48, -1.98, 5.86, 5.57, 4.57, 9.75, 9.97,
     10.90, 9.19, 2.93, 5.10, -1.52, -3.93, -1.95, -2.46, -0.64, 4.60, -1.82, -0.80,
     9.34, 7.51, 6.45, 5.23, 4.41, 3.60, -1.94, -1.15),
```

```
  p_val_adj = c(
    3.82e-09, 1.52e-07, 1.79e-07, 4.68e-07, 4.83e-07, 6.26e-07, 2.61e-06, 1.33e-05,
    1.79e-05, 3.71e-05, 5.21e-05, 5.36e-05, 5.83e-05, 6.66e-05, 8.22e-05, 2.89e-04,
    3.00e-04, 4.94e-04, 7.62e-04, 8.93e-04, 9.55e-04, 9.61e-04, 1.12e-03, 1.47e-03,
    1.66e-03, 1.95e-03, 2.06e-03, 3.01e-03, 3.26e-03, 4.35e-03, 4.85e-03, 5.12e-03,
    5.40e-03, 7.18e-03, 7.18e-03, 1.04e-02, 1.24e-02, 1.90e-02, 1.94e-02, 1.97e-02,
    2.09e-02, 2.13e-02, 2.25e-02, 2.61e-02, 3.18e-02, 3.27e-02, 3.69e-02, 3.80e-02,
    4.95e-02, 5.73e-02, 5.77e-02, 6.10e-02, 6.22e-02, 6.31e-02, 6.72e-02, 9.23e-02,
    9.85e-02, 1.06e-01, 1.07e-01, 1.11e-01, 1.31e-01, 1.38e-01, 1.40e-01, 1.43e-01,
    2.00e-01, 2.39e-01, 2.49e-01, 2.57e-01, 2.86e-01, 2.86e-01, 2.98e-01, 3.32e-01,
    4.15e-01, 4.91e-01, 4.91e-01, 4.91e-01, 5.97e-01, 7.11e-01, 7.59e-01, 8.38e-01,
    9.20e-01, 9.20e-01, 9.29e-01, 9.29e-01, 9.29e-01, 9.29e-01, 9.34e-01, 9.68e-01,
    1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00,
    1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00, 1.00e+00),
  gene = c(
    "HLA-DPB1", "LYZ", "HLA-DRA", "TYMP", "HLA-DPA1", "HLA-DRB1", "CST3", "HLA-DQB1",
    "HLA-DRB5", "LST1", "HLA-DQA1", "AIF1", "S100A8", "IFITM3", "HLA-DMB", "FCGRT",
    "SERPINA1", "IFI30", "S100A9", "CCL5", "GRN", "LCK", "HLA-DMA", "MS4A6A", "CTSS",
    "CFP", "FCN1", "BID", "CFD", "CD3D", "CD7", "CD3E", "LGALS2", "CD14", "SMCO4",
    "LINC00936", "HCK", "CTSW", "LGALS1", "HLA-DQA2", "LRRC25", "GZMM", "RNF130",
    "LGALS3", "S100A11", "C5AR1", "IL1B", "GZMA", "FCER1G", "MPEG1", "TYROBP", "TSPO",
    "GSTP1", "CTSB", "IL32", "CD247", "GNLY", "COTL1", "NFKBIA", "NUP214", "LAMP1",
    "FPR1", "CLEC10A", "CST7", "PRF1", "BLVRA", "PSAP", "GZMH", "EAF2", "ASGR1",
   "RARRES3", "SAT1", "LY86", "GP9", "TUBB1", "NGFRAP1", "XBP1", "SCO2", "RGS2", "GZMB",
    "HIST1H2AC", "KLRD1", "PGRMC1", "AKR1C3", "PTGDR", "IL2RB", "GYPC", "CCL4", "CD68",
    "FCER1A", "CD79B", "MS4A7", "CARD16", "ACAP1", "CD79A", "ANXA2", "TMEM40", "PF4",
    "GNG11", "CLU", "CD9", "FGFBP2", "TNFRSF1B", "IFI6"),
  pct_diff = c(
    -0.752, -0.457, -0.460, -0.671, -0.626, -0.701, -0.502, -0.619, -0.623, -0.598,
    -0.566, -0.626, -0.543, -0.566, -0.541, -0.542, -0.515, -0.489, -0.444, 0.428,
    -0.517, 0.461, -0.491, -0.410, -0.480, -0.491, -0.521, -0.491, -0.438, 0.411,
   0.411, 0.409, -0.438, -0.359, -0.359, -0.440, -0.386, 0.385, -0.332, -0.361, -0.361,
   0.364, -0.387, -0.415, -0.454, -0.308, -0.335, 0.364, -0.454, -0.309, -0.379, -0.427,
    -0.377, -0.389, 0.335, 0.315, 0.313, -0.284, -0.502, -0.309, 0.313, -0.284, -0.256,
    0.309, 0.313, -0.364, -0.406, 0.244, -0.231, -0.231, 0.281, -0.311, -0.312, 0.220,
   0.220, 0.220, 0.261, -0.232, -0.367, 0.240, 0.218, 0.218, 0.195, 0.195, 0.195, 0.195,
    0.262, 0.218, -0.288, -0.207, -0.290, -0.233, -0.367, 0.217, -0.233, -0.403, 0.171,
    0.194, 0.194, 0.194, 0.194, 0.213, -0.235, -0.292),
  group = sample(LETTERS[1:2], 104, replace = TRUE)
)
# If set, it will be used as labels if label_by is not set.
# rownames(data) <- data$gene

VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)")
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)", label_by = "gene")
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE, label_by = "gene")
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE, facet_by = "group", label_by = "gene")
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   flip_negatives = TRUE, split_by = "group", label_by = "gene")
```

```
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", y_cutoff_name = "none",
   highlight = c("ANXA2", "TMEM40", "PF4", "GNG11", "CLU", "CD9", "FGFBP2",
   "TNFRSF1B", "IFI6"), label_by = "gene")
VolcanoPlot(data, x = "avg_log2FC", y = "p_val_adj", color_by = "pct_diff",
   y_cutoff_name = "-log10(0.05)", split_by = "group", label_by = "gene",
   palette = c(A = "Set1", B = "Dark2"))
```

`WordCloudPlot` *Word Cloud Plot*

Description

Word cloud plot to illustrate the count/frequency of words.

Usage

```
WordCloudPlot(
  data,
  word_by = NULL,
  sentence_by = NULL,
  count_by = NULL,
  score_by = NULL,
  count_name = NULL,
  score_name = NULL,
  split_by = NULL,
  split_by_sep = "_",
  words_excluded = plotthis::words_excluded,
  score_agg = mean,
  minchar = 2,
  word_size = c(2, 8),
  top_words = 100,
  facet_by = NULL,
  facet_scales = "fixed",
  facet_ncol = NULL,
  facet_nrow = NULL,
  facet_byrow = TRUE,
  theme = "theme_this",
  theme_args = list(),
  palette = "Spectral",
  palcolor = NULL,
  alpha = 1,
  palreverse = FALSE,
  aspect.ratio = 1,
  legend.position = "right",
  legend.direction = "vertical",
  title = NULL,
  subtitle = NULL,
```

```
  seed = 8525,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  axes = NULL,
  axis_titles = axes,
  guides = NULL,
  design = NULL,
  ...
)
```

Arguments

data	A data frame.
word_by	A character string of the column name to use as the word. A character column is expected.
sentence_by	A character string of the column name to split the sentence. A character column is expected. Either word_by or sentence_by should be specified.
count_by	A character string of the column name for the count of the word/sentence. A numeric column is expected. If NULL, the count of the word/sentence will be used.
score_by	A character string of the column name for the score of the word/sentence. A numeric column is expected, used for the color of the word cloud. If NULL, the score will be set to 1.
count_name	A character string to name the legend of count.
score_name	A character string to name the legend of score.
split_by	The column(s) to split data by and plot separately.
split_by_sep	The separator for multiple split_by columns. See split_by
words_excluded	A character vector of words to exclude from the word cloud.
score_agg	A function to aggregate the scores. Default is mean.
minchar	A numeric value specifying the minimum number of characters for the word.
word_size	A numeric vector specifying the range of the word size.
top_words	A numeric value specifying the number of top words to show.
facet_by	A character string specifying the column name of the data frame to facet the plot. Otherwise, the data will be split by split_by and generate multiple plots and combine them into one using patchwork::wrap_plots
facet_scales	Whether to scale the axes of facets. Default is "fixed" Other options are "free", "free_x", "free_y". See ggplot2::facet_wrap
facet_ncol	A numeric value specifying the number of columns in the facet. When facet_by is a single column and facet_wrap is used.
facet_nrow	A numeric value specifying the number of rows in the facet. When facet_by is a single column and facet_wrap is used.

facet_byrow — A logical value indicating whether to fill the plots by row. Default is TRUE.

theme — A character string or a theme class (i.e. ggplot2::theme_classic) specifying the theme to use. Default is "theme_this".

theme_args — A list of arguments to pass to the theme function.

palette — A character string specifying the palette to use. A named list or vector can be used to specify the palettes for different split_by values.

palcolor — A character string specifying the color to use in the palette. A named list can be used to specify the colors for different split_by values. If some values are missing, the values from the palette will be used (palcolor will be NULL for those values).

alpha — A numeric value specifying the transparency of the plot.

palreverse — A logical value indicating whether to reverse the palette. Default is FALSE.

aspect.ratio — A numeric value specifying the aspect ratio of the plot.

legend.position — A character string specifying the position of the legend. if waiver(), for single groups, the legend will be "none", otherwise "right".

legend.direction — A character string specifying the direction of the legend.

title — A character string specifying the title of the plot. A function can be used to generate the title based on the default title. This is useful when split_by is used and the title needs to be dynamic.

subtitle — A character string specifying the subtitle of the plot.

seed — The random seed to use. Default is 8525.

combine — Whether to combine the plots into one when facet is FALSE. Default is TRUE.

nrow — A numeric value specifying the number of rows in the facet.

ncol — A numeric value specifying the number of columns in the facet.

byrow — A logical value indicating whether to fill the plots by row.

axes — A string specifying how axes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axes in individual plots.
- 'collect' will remove duplicated axes when placed in the same run of rows or columns of the layout.
- 'collect_x' and 'collect_y' will remove duplicated x-axes in the columns or duplicated y-axes in the rows respectively.

axis_titles — A string specifying how axis titltes should be treated. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'keep' will retain all axis titles in individual plots.
- 'collect' will remove duplicated titles in one direction and merge titles in the opposite direction.
- 'collect_x' and 'collect_y' control this for x-axis titles and y-axis titles respectively.

guides — A string specifying how guides should be treated in the layout. Passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. Options are:

- 'collect' will collect guides below to the given nesting level, removing duplicates.
- 'keep' will stop collection at this level and let guides be placed alongside their plot.
- 'auto' will allow guides to be collected if a upper level tries, but place them alongside the plot if not.

design — Specification of the location of areas in the layout, passed to patchwork::wrap_plots(). Only relevant when split_by is used and combine is TRUE. When specified, nrow, ncol, and byrow are ignored. See patchwork::wrap_plots() for more details.

... — Additional arguments.

Value

A ggplot object or wrap_plots object or a list of ggplot objects

Examples

```
data <- data.frame(
   word = c("apple", "banana", "cherry", "date", "elderberry"),
   count = c(10, 20, 30, 40, 50),
   score = c(1, 2, 3, 4, 5)
)
WordCloudPlot(data, word_by = "word", count_by = "count", score_by = "score")
```

---

words_excluded *Excluded words in keyword enrichment analysis and extraction*

Description

The variable "words_excluded" represents the words that are excluded during keyword enrichment analysis or keyword extraction process. These mainly include words that are excessively redundant or of little value.

Examples

```
## Not run:
if (interactive()) {
  words_excluded <- c(
    "the", "is", "and", "or", "a", "in", "on", "under", "between", "of", "through",
    "via", "along", "that", "for", "with", "within", "without", "cell", "cellular",
    "dna", "rna", "protein", "peptide", "amino", "acid", "development", "involved",
    "organization", "system", "regulation", "regulated", "positive", "negative",
    "response", "process", "processing", "small", "large", "change", "disease"
  )
```

```
}

## End(Not run)
```

Index