Package 'thisplot'

October 28, 2026

Type Package

Title Utility Functions for Plotting

Version 0.4.6

Date 2026-10-28

Maintainer Meng Xu <mengxu98@qq.com>

Description Provides utility functions for plotting. Includes functions for color manipulation, plot customization, panel size control, data optimization for plots, and layout adjustments.

License MIT + file LICENSE

URL https://mengxu98.github.io/thisplot/

BugReports https://github.com/mengxu98/thisplot/issues

Depends R (>= 4.1.0)

Imports cli, dendextend, ggforce, geomtextpath, ggrepel, ggnewscale, ggplot2 (>= 4.0.0), grDevices, grid, gtable, igraph, methods, patchwork, proxyC, reshape2, stats, thisutils (>= 0.5.0), utils

Suggests concaveman, circlize, ComplexHeatmap, dplyr, forcats, ggalluvial, ggExtra, ggpointdensity, ggrastr, ggupset, ggVennDiagram, htmltools, Matrix, png, purrr, ragg, scales, stringr, tidyr, venny

Config/Needs/website mengxu98/thistemplate

Encoding UTF-8

Language en-US

LazyData true

Config/roxygen2/version 8.1.0

RoxygenNote 8.1.1

NeedsCompilation no

Author Meng Xu [aut, cre] (ORCID: <https://orcid.org/0000-0002-8300-1054>), Haoliang Zhu [aut] (ORCID: <https://orcid.org/0009-0004-9766-5179>)

Repository CRAN

Date/Publication 2026-10-28 08:40:02 UTC

Contents

thisplot-package . . . 3
add_grob . . . 4
adjcolors . . . 5
adjustlayout . . . 5
annotate_quadrants . . . 6
annotation_block_fill_graphics . . . 7
annotation_block_graphics . . . 8
annotation_graphics . . . 8
as_grob . . . 9
as_gtable . . . 9
Blend2Color . . . 10
blendcolors . . . 10
BlendRGBList . . . 11
build_heatmap_annotation . . . 12
build_patchwork . . . 13
ChineseColors . . . 13
chinese_colors . . . 15
clip_symmetric_range . . . 16
cluster_within_group2 . . . 16
col2hex . . . 17
compute_velocity_on_grid . . . 17
DimDataPlot . . . 19
drop_data . . . 23
extractgrobs . . . 24
geom_alluvial . . . 24
geom_alluvial_label . . . 25
geom_sankey . . . 27
geom_sankey_bump . . . 28
geom_sankey_label . . . 29
get_chinese_palettes . . . 30
get_colors . . . 31
get_legend . . . 32
get_vars . . . 33
GraphPlot . . . 33
grid_draw . . . 38
head.colors . . . 38
heatmap_border_spec . . . 39
heatmap_fixsize . . . 40
heatmap_rendersize . . . 41
invert_figures . . . 42
invert_png . . . 43
invert_svg . . . 43
jitter_highlighted_points . . . 44
major_grid . . . 45
make_long . . . 46
mestimate . . . 46

NetworkPlot 47
normalize_drawable 50
no_legend 50
no_margin 51
no_x_text 51
no_y_text 52
palette_colors 52
palette_list 54
panel_fix 54
patchwork_grob 57
plot_autocompose 57
plot_autosize 59
plot_histogram 60
plot_scatter 61
print.ChineseColors 63
print.colors 63
print.thisplot_logo 64
print_colored_table 64
rasterise_plot 65
RGBA2RGB 66
segements_df 67
show_palettes 68
simple_colors 69
slim_data 70
standardise 71
StatPlot 71
theme_blank 77
theme_sankey 79
theme_this 79
thisplot_logo 80
to_percent 81
visual_colors 81

Index **83**

`thisplot-package` *Utility Functions for Data Visualization and Plotting*

Description

Provides utility functions for data visualization and plotting in R. Includes functions for color manipulation, plot customization, panel size control, data optimization for plots, and layout adjustments. Designed to enhance workflows with ggplot2, patchwork, and ComplexHeatmap.

Author(s)

Meng Xu (Maintainer), <mengxu98@qq.com>

Source

https://mengxu98.github.io/thisplot/

See Also

Useful links:

- https://mengxu98.github.io/thisplot/
- Report bugs at https://github.com/mengxu98/thisplot/issues

`add_grob` *Add a grob to a gtable*

Description

Add a grob to a gtable at a specified position (top, bottom, left, or right).

Usage

```
add_grob(
  gtable,
  grob,
  position = c("top", "bottom", "left", "right", "none"),
  space = NULL,
  clip = "on"
)
```

Arguments

`gtable`	A gtable object.
`grob`	A grob or gtable object to add.
`position`	The position to add the grob. One of `"top"`, `"bottom"`, `"left"`, `"right"`, or `"none"`.
`space`	The space to allocate for the grob. If `NULL`, will be calculated automatically.
`clip`	The clipping mode. Default is `"on"`.

Value

A gtable object with the grob added.

adjcolors *Convert a color with specified alpha level*

Description

Convert a color with specified alpha level

Usage

```
adjcolors(colors, alpha)
```

Arguments

colors	Color vectors.
alpha	Alpha level in [0,1].

Value

A character vector of hexadecimal color codes with the specified alpha level.

Examples

```
colors <- c("red", "blue", "green")
adjcolors(colors, 0.5)
ggplot2::alpha(colors, 0.5)

show_palettes(
  list(
    "raw" = colors,
    "adjcolors" = adjcolors(colors, 0.5),
    "ggplot2::alpha" = ggplot2::alpha(colors, 0.5)
  )
)
```

adjustlayout *Adjust graph layout to avoid node overlaps*

Description

Adjust the layout of a graph to prevent node overlaps by considering node widths and heights.

Usage

```
adjustlayout(graph, layout, width, height = 2, scale = 100, iter = 100)
```

Arguments

graph	An igraph graph object.
layout	A matrix with two columns representing the initial layout coordinates.
width	A numeric vector of node widths.
height	The height constraint for nodes. Default is 2.
scale	The scaling factor for the layout. Default is 100.
iter	The number of iterations for the adjustment algorithm. Default is 100.

Value

A matrix with adjusted layout coordinates.

---

annotate_quadrants *Annotate plot quadrants with percentages*

Description

Annotate plot quadrants with percentages

Usage

```
annotate_quadrants(
  plot,
  x,
  y,
  cutoffs,
  line_color = "grey30",
  line_type = "solid",
  line_width = 0.5,
  label_size = 3,
  group = NULL
)
```

Arguments

plot	A ‘ggplot‘ object.
x	Name of the x variable in ‘plot$data‘.
y	Name of the y variable in ‘plot$data‘.
cutoffs	Cut points for x and y axes. Can be a numeric vector of length 1 or 2, or a list with the x and y cut points.
line_color	Quadrant line color.
line_type	Quadrant line type.
line_width	Quadrant line width.
label_size	Label size.
group	Optional grouping column used to compute percentages within each group.

Value

A 'ggplot' object.

Examples

```
df <- data.frame(
  x = c(0.1, 0.2, 0.8, 0.9),
  y = c(0.1, 0.8, 0.2, 0.9)
)
p <- ggplot2::ggplot(df, ggplot2::aes(x = x, y = y)) +
  ggplot2::geom_point()
p

annotate_quadrants(p, x = "x", y = "y", cutoffs = 0.5)
```

`annotation_block_fill_graphics` *Build block fill panel function*

Description

Build block fill panel function

Usage

```
annotation_block_fill_graphics(
  levels,
  palette = NULL,
  palcolor = NULL,
  fill_values = NULL,
  border = TRUE,
  border_color = "black",
  border_size = 1
)
```

Arguments

`levels`	Character/factor levels used as keys for fill mapping.
`palette`	Palette name. All available palette names can be queried with show_palettes.
`palcolor`	Custom colors used to create a color palette.
`fill_values`	Optional named fill vector. Names should match `levels`.
`border`	Whether to draw block border.
`border_color`	Border color when `border = TRUE`. Default is `"black"`.
`border_size`	Border line width when `border = TRUE`. Default is 1.

Value

A function with signature `(index, levels)` suitable for `panel_fun` in ComplexHeatmap::anno_block.

annotation_block_graphics *Build block panel function*

Description

Build block panel function

Usage

```
annotation_block_graphics(
  subplot,
  name,
  border = TRUE,
  border_color = "black",
  border_size = 1
)
```

Arguments

subplot	A single drawable subplot object.
name	A name assigned to the generated grob.
border	Whether to draw a rectangle border around the block.
border_color	Border color when border = TRUE. Default is "black".
border_size	Border line width when border = TRUE. Default is 1.

Value

A function with signature (index, levels) suitable for panel_fun in ComplexHeatmap::anno_block.

annotation_graphics *Build graphics callback list for anno_customize*

Description

Build graphics callback list for anno_customize

Usage

```
annotation_graphics(subplots, prefix)
```

Arguments

subplots	A named list of subplot objects.
prefix	Prefix used to generate stable grob names.

Value

A named list of callback functions accepted by ComplexHeatmap::anno_customize.

`as_grob` *Convert a plot object to a grob*

Description

Convert various plot objects (gList, patchwork, ggplot) to a grob object.

Usage

```
as_grob(plot, ...)
```

Arguments

`plot`	A plot object (gList, patchwork, or ggplot).
`...`	Additional arguments passed to other functions.

Value

A grob object.

`as_gtable` *Convert a plot object to a gtable*

Description

Convert various plot objects (gtable, grob, patchwork, ggplot) to a gtable object.

Usage

```
as_gtable(plot, ...)
```

Arguments

`plot`	A plot object (gtable, grob, patchwork, or ggplot).
`...`	Additional arguments passed to other functions.

Value

A gtable object.

Blend2Color *Blend two colors using a specified mode*

Description

Blend two colors with alpha channels using one of several blending modes: blend, average, screen, or multiply.

Usage

```
Blend2Color(C1, C2, mode = "blend")
```

Arguments

C1	A list containing the first color RGB values and alpha channel.
C2	A list containing the second color RGB values and alpha channel.
mode	The blending mode to use. One of "blend", "average", "screen", or "multiply". Default is "blend".

Value

A list containing the blended RGB values and alpha channel.

blendcolors *Blends a list of colors using the specified blend mode*

Description

Blends a list of colors using the specified blend mode

Usage

```
blendcolors(colors, mode = c("blend", "average", "screen", "multiply"))
```

Arguments

colors	Color vectors.
mode	Blend mode. One of "blend", "average", "screen", or "multiply".

Value

A character vector of hexadecimal color codes representing the blended color.

Examples

```
blend <- c(
  "red",
  "green",
  blendcolors(c("red", "green"),
    mode = "blend"
  )
)
average <- c(
  "red",
  "green",
  blendcolors(c("red", "green"),
    mode = "average"
  )
)
screen <- c(
  "red",
  "green",
  blendcolors(c("red", "green"),
    mode = "screen"
  )
)
multiply <- c(
  "red",
  "green",
  blendcolors(c("red", "green"),
    mode = "multiply"
  )
)
show_palettes(
  list(
    "blend" = blend,
    "average" = average,
    "screen" = screen,
    "multiply" = multiply
  )
)
```

`BlendRGBList` *Blend a list of colors*

Description

Blend multiple colors with alpha channels into a single color using a specified blending mode.

Usage

```
BlendRGBList(Clist, mode = "blend", RGB_BackGround = c(1, 1, 1))
```

Arguments

Clist	A list of colors, where each color is a list containing RGB values and alpha channel.
mode	The blending mode to use. One of "blend", "average", "screen", or "multiply". Default is "blend".
RGB_BackGround	The background RGB color to composite with. Default is c(1, 1, 1) (white).

Value

A numeric vector of RGB values.

---

build_heatmap_annotation

Build HeatmapAnnotation with safe parameter merge

Description

Build HeatmapAnnotation with safe parameter merge

Usage

```
build_heatmap_annotation(
  annotations,
  which,
  show_annotation_name = TRUE,
  annotation_name_side = NULL,
  border = NULL,
  params = NULL
)
```

Arguments

annotations	Named list of annotation components (for example, anno_simple, anno_block, anno_customize).
which	Annotation direction ("row" or "column").
show_annotation_name	Whether to show annotation names.
annotation_name_side	Side for annotation names.
border	Whether to draw a rectangle border around the block.
params	Additional user parameters; duplicated keys are ignored if already set explicitly.

Value

A ComplexHeatmap::HeatmapAnnotation object.

build_patchwork *Build a patchwork gtable*

Description

Build a gtable from a patchwork object by arranging multiple plots according to the layout specification.

Usage

```
build_patchwork(
  x,
  guides = "auto",
  table_rows = 18,
  table_cols = 15,
  panel_row = 10,
  panel_col = 8
)
```

Arguments

x	A patchwork object.
guides	How to handle guides. Default is "auto".
table_rows	The number of rows in the table grid. Default is 18.
table_cols	The number of columns in the table grid. Default is 15.
panel_row	The row index for panels. Default is 10.
panel_col	The column index for panels. Default is 8.

Value

A gtable object.

ChineseColors *Chinese traditional colors system*

Description

A color system based on Chinese traditional colors with 1058 colors.

Usage

```
ChineseColors()
```

Value

A `ChineseColors` object. Detailed information can be found in `print.ChineseColors()`.

See Also

chinese_colors for the dataset of Chinese traditional colors. get_chinese_palettes for getting Chinese color palettes. visual_colors for visualizing any color vector. get_colors for searching colors in dataset and palettes.

Examples

```
cc <- ChineseColors()
cc

# Get a color by pinyin
get_colors("pinlan")

# By number
get_colors(44)

# By hex code
get_colors("#2B73AF")

# Multiple colors
get_colors("pinlan", "piao")
get_colors(91:100)

# Chinese names
cc$visual_colors(
  loc_range = c(1, 180),
  num_per_row = 30,
  title = "Chinese Traditional Colors",
  name_type = "chinese"
)

# pinyin as names
cc$visual_colors(
  loc_range = c(1, 120),
  title = "Chinese Traditional Colors",
  name_type = "pinyin"
)

# rgb as names
cc$visual_colors(
  loc_range = c(1, 120),
  title = "Colors with RGB values",
  name_type = "rgb"
)

# hex as names
cc$visual_colors(
  loc_range = c(1, 120),
```

```
  title = "Colors with hex codes",
  name_type = "hex"
)
```

`chinese_colors` *Chinese traditional colors dataset*

Description

A dataset containing optimized traditional Chinese colors. These colors are extracted from those books:

- Chinese Traditional Colors - Color Aesthetics in the Forbidden City (ISBN: 9787521716054)
- Chinese Beautiful Colors - The Most Chinese Culture Vol.3 (ISBN: 9781672897198)
- Chinese Colors (ISBN: 9787558016479)
- Chinese Journal of Chromatography (ISSN 1000-8713)
- Chinese Color Atlas

Thanks to the author of the blog for providing the data.

Examples

```
data(chinese_colors)
color_sets <- attr(chinese_colors, "color_sets")
show_palettes(
  list(
    color_sets$Chinese,
    color_sets$ChineseSet8,
    color_sets$ChineseSet16,
    color_sets$ChineseSet32
  )
)

# Use ChineseColors class
cc <- ChineseColors()
cc$visual_colors(
  loc_range = c(1, 180),
  num_per_row = 30,
  title = "Chinese Traditional Colors",
  name_type = "chinese"
)
```

clip_symmetric_range *Clip values to a symmetric range*

Description

Clip values to a symmetric range

Usage

```
clip_symmetric_range(data, value_col = "avg_log2FC")
```

Arguments

data	A data frame.
value_col	Name of the numeric column to clip.

Value

A list containing the clipped data frame and clipping limits.

Examples

```
df <- data.frame(avg_log2FC = c(-10, -2, 0, 2, 10))
clip_symmetric_range(df)
```

cluster_within_group2 *Cluster within group*

Description

Cluster within group

Usage

```
cluster_within_group2(mat, factor)
```

Arguments

mat	A matrix of data
factor	A factor

Value

A dendrogram with ordered leaves

col2hex *Convert color names to hexadecimal format*

Description

Convert color names to hexadecimal RGB color codes.

Usage

```
col2hex(cname)
```

Arguments

cname	A character vector of color names.

Value

A character vector of hexadecimal color codes.

compute_velocity_on_grid *Compute velocity on grid*

Description

Compute velocity on grid

Usage

```
compute_velocity_on_grid(
  x_emb,
  v_emb,
  density = 1,
  smooth = 0.5,
  n_neighbors = ceiling(n_obs/50),
  min_mass = 1,
  scale = 1,
  adjust_for_stream = FALSE,
  cutoff_perc = 5
)
```

Arguments

x_emb	A matrix of dimension n_obs x n_dim specifying the embedding coordinates of the cells.
v_emb	A matrix of dimension n_obs x n_dim specifying the velocity vectors of the cells.
density	A numeric value specifying the density of the grid points along each dimension. Default is 1.
smooth	A numeric value specifying the smoothing factor for the velocity vectors. Default is 0.5.
n_neighbors	A numeric value specifying the number of nearest neighbors for each grid point. Default is ceiling(n_obs / 50).
min_mass	A numeric value specifying the minimum mass required for a grid point to be considered. Default is 1.
scale	A numeric value specifying the scaling factor for the velocity vectors. Default is 1.
adjust_for_stream	Whether to adjust the velocity vectors for streamlines. Default is FALSE.
cutoff_perc	A numeric value specifying the percentile cutoff for removing low-density grid points. Default is 5.

Value

A list with two components: x_grid, the grid coordinates, and v_grid, the smoothed velocity vectors on the grid.

References

https://github.com/theislab/scvelo/blob/master/scvelo/plotting/velocity_embedding_grid.py

Examples

```
x_emb <- matrix(
  c(
    0, 0,
    1, 0,
    0, 1,
    1, 1
  ),
  ncol = 2,
  byrow = TRUE
)
v_emb <- matrix(
  c(
    1, 0,
    1, 0,
    0, 1,
    0, 1
```

```
  ),
  ncol = 2,
  byrow = TRUE
)

velocity_grid <- compute_velocity_on_grid(
  x_emb = x_emb,
  v_emb = v_emb,
  density = 0.1,
  n_neighbors = 2,
  adjust_for_stream = TRUE
)

names(velocity_grid)
dim(velocity_grid$x_grid)
dim(velocity_grid$v_grid)
head(velocity_grid$x_grid)
head(velocity_grid$v_grid)

grid_df <- expand.grid(
  x = velocity_grid$x_grid[1, ],
  y = velocity_grid$x_grid[2, ]
)

plot_df <- data.frame(
  x = grid_df$x,
  y = grid_df$y,
  xend = grid_df$x + c(velocity_grid$v_grid[1, , ]) * 0.2,
  yend = grid_df$y + c(velocity_grid$v_grid[2, , ]) * 0.2
)

ggplot2::ggplot(plot_df) +
  ggplot2::geom_segment(
    ggplot2::aes(x = x, y = y, xend = xend, yend = yend),
    arrow = grid::arrow(length = grid::unit(0.12, "inches")),
    na.rm = TRUE
  ) +
  ggplot2::coord_equal()
```

`DimDataPlot` *Data Dimensional Plot*

Description

Visualize observations on precomputed two-dimensional coordinates.

Usage

```
DimDataPlot(
  data,
```

```
  x,
  y,
  group.by = NULL,
  split.by = NULL,
  shape.by = NULL,
  show_na = FALSE,
  palette = "Chinese",
  palcolor = NULL,
  NA_color = "grey80",
  pt.size = 2,
  pt.alpha = 1,
  shape.values = NULL,
  add_mark = FALSE,
  mark_type = c("hull", "ellipse", "rect", "circle"),
  mark_expand = grid::unit(3, "mm"),
  mark_alpha = 0.1,
  mark_linetype = 1,
  label = FALSE,
  label.size = 4,
  label.fg = "black",
  label.bg = "white",
  label.bg.r = 0.1,
  label_repel = FALSE,
  label_repulsion = 20,
  label_point_size = 1,
  label_point_color = "black",
  label_segment_color = "black",
  add_origin = FALSE,
  origin.color = "grey30",
  origin.linetype = "dashed",
  origin.linewidth = 0.4,
  aspect.ratio = 1,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  legend.title = NULL,
  theme_use = "theme_this",
  theme_args = list(),
  seed = 11
)
```

Arguments

data	A data frame containing coordinates and optional grouping columns.
x	Name of the column used as the x-axis coordinate.

y	Name of the column used as the y-axis coordinate.
group.by	Name of the column used to color observations. Default is NULL.
split.by	Name of the column used to split the plot into facets. Default is NULL.
shape.by	Name of the column used to map point shapes. Default is NULL.
show_na	Whether to keep missing values in group.by, split.by, or shape.by as an explicit "NA" level. Default is FALSE.
palette	Color palette name. Available palettes can be found in show_palettes. Default is "Chinese".
palcolor	Custom colors used to create a color palette. Default is NULL.
NA_color	Color used for missing values.
pt.size	Point size.
pt.alpha	Point transparency.
shape.values	Point shapes used when shape.by is set.
add_mark	Whether to add marks around groups. Default is FALSE.
mark_type	Type of mark to add around groups. One of "hull", "ellipse", "rect", or "circle". Default is "hull". "ellipse" uses ggplot2::stat_ellipse with type = "norm" and level = 0.95; the other mark types use ggforce marks.
mark_expand	Expansion of the mark around each group. This is used for "hull", "rect", and "circle" marks. Default is grid::unit(3, "mm").
mark_alpha	Transparency of the mark. Default is 0.1.
mark_linetype	Line type of the mark border. Default is 1.
label	Whether to label group centers.
label.size	Size of labels.
label.fg	Foreground color of labels.
label.bg	Background color of labels.
label.bg.r	Background ratio of labels.
label_repel	Whether labels repel from their group centers.
label_repulsion	Repulsion force for labels.
label_point_size	Size of center points for repelled labels.
label_point_color	Color of center points for repelled labels.
label_segment_color	Color of label segments.
add_origin	Whether to add dashed x = 0 and y = 0 reference lines.
origin.color	Color of origin reference lines.
origin.linetype	Line type of origin reference lines.
origin.linewidth	Line width of origin reference lines.

aspect.ratio	Aspect ratio passed to theme().
title	Plot title.
subtitle	Plot subtitle.
xlab	Label for the x axis.
ylab	Label for the y axis.
legend.position	Position of the legend.
legend.direction	Direction of the legend.
legend.title	Title of the group legend.
theme_use	Theme function applied to the plot.
theme_args	A named list of arguments passed to theme_use.
seed	Random seed.

Value

A ggplot object.

Examples

```
set.seed(1)
plot_data <- data.frame(
  x = c(rnorm(40, -1), rnorm(40, 1)),
  y = c(rnorm(40, 0), rnorm(40, 1)),
  group = rep(c("A", "B"), each = 40)
)
DimDataPlot(
  plot_data,
  x = "x",
  y = "y",
  group.by = "group"
)

DimDataPlot(
  plot_data,
  x = "x",
  y = "y",
  group.by = "group",
  add_origin = TRUE,
  mark_type = "ellipse"
)

DimDataPlot(
  plot_data,
  x = "x",
  y = "y",
  group.by = "group",
  mark_type = "circle",
```

```
  mark_alpha = 0.3,
  mark_expand = grid::unit(1, "mm"),
  mark_linetype = 2
)
```

---

drop_data *Drop unused data in the plot*

Description

Drop unused data points from a ggplot or patchwork object while preserving the plot structure. This function keeps only a single row of data for each unique combination of used variables, which can significantly reduce the object size when the original data contains many rows that are not displayed in the plot (e.g., due to scale limits or filtering).

Usage

```
drop_data(p)

## S3 method for class 'ggplot'
drop_data(p)

## S3 method for class 'patchwork'
drop_data(p)

## Default S3 method:
drop_data(p)
```

Arguments

p	A ggplot object or a patchwork object.

Value

A ggplot or patchwork object with unused data points removed.

Examples

```
library(ggplot2)
library(patchwork)
p <- ggplot(
  data = mtcars,
  aes(x = mpg, y = wt, colour = cyl)
) +
  geom_point() +
  scale_x_continuous(limits = c(10, 30)) +
  scale_y_continuous(limits = c(1, 6))
object.size(p)
```

```
p_drop <- drop_data(p)
object.size(p_drop)

p / p_drop
```

`extractgrobs` *Extract grobs from a list*

Description

Extract grobs from a named list of grobs based on the specified x and y indices.

Usage

```
extractgrobs(vlnplots, x_nm, y_nm, x, y)
```

Arguments

`vlnplots`	A named list of grobs.
`x_nm`	A character vector of names for the x dimension.
`y_nm`	A character vector of names for the y dimension.
`x`	An integer index for the x dimension.
`y`	An integer index for the y dimension.

Value

The extracted grob(s).

`geom_alluvial` *geom_alluvial*

Description

Creates an alluvial plot which visualize flows between nodes. Each observation needs to have a ‘x‘ aesthetic as well as a ‘next_x‘ column which declares where that observation should flow. Also each observation should have a ‘node‘ and a ‘next_node‘ aesthetic which provide information about which group in the y-direction.

Usage

```
geom_alluvial(
  mapping = NULL,
  data = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  space = 0,
  width = 0.1,
  smooth = 8,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Provide you own mapping. Both x and y need to be numeric.
data	Provide you own data.
position	Change position.
na.rm	Remove missing values.
show.legend	Show legend in plot.
space	Space between nodes in the y-direction.
width	Width of nodes.
smooth	How much smooth should the curve have? More means steeper curve.
inherit.aes	Should the geom inherit aesthetics.
...	Other arguments to be passed to the geom.

Value

A ggplot layer.

---

geom_alluvial_label *geom_alluvial_label*

Description

Creates centered labels or text in nodes of your alluvial plot. Needs to have the exact same aesthetics as the call to ‘geom_alluvial‘ to work.

Usage

```
geom_alluvial_text(
  mapping = NULL,
  data = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  space = 0,
  width = 0.1,
  inherit.aes = TRUE,
  ...
)

geom_alluvial_label(
  mapping = NULL,
  data = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  space = 0,
  width = 0.1,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Provide you own mapping. Both x and y need to be numeric.
data	Provide you own data.
position	Change position.
na.rm	Remove missing values.
show.legend	Show legend in plot.
space	Space between nodes in the y-direction.
width	Width of nodes.
inherit.aes	Should the geom inherit aesthetics.
...	Other arguments to be passed to the geom.

Details

Other important arguments are; ‘space‘ which provides the space between nodes in the y-direction; ‘shift‘ which shifts nodes in the y-direction.

Value

A ggplot layer.

geom_sankey *geom_sankey*

Description

Creates a sankey plot which visualize flows between nodes. Each observation needs to have a ‘x‘ aesthetic as well as a ‘next_x‘ column which declares where that observation should flow. Also each observation should have a ‘node‘ and a ‘next_node‘ aesthetic which provide information about which group in the y-direction. By default each row of the data frame is counted to calculate the size of flows. A manual flow value can be added with the ‘value‘ aesthetic.

Usage

```
geom_sankey(
  mapping = NULL,
  data = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  space = NULL,
  type = "sankey",
  width = 0.1,
  smooth = 8,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Provide you own mapping. Both x and y need to be numeric.
data	Provide you own data.
position	Change position.
na.rm	Remove missing values.
show.legend	Show legend in plot.
space	Space between nodes in the y-direction.
type	Either ‘sankey‘ or ‘alluvial‘.
width	Width of nodes.
smooth	How much smooth should the curve have? More means steeper curve.
inherit.aes	Should the geom inherit aesthetics.
...	Other arguments to be passed to the geom.

Value

A ggplot layer.

Aesthetics

geom_sankey understand the following aesthetics (required aesthetics are in bold):

- **x0** - **y0** - **a** - **b** - **angle** - m1 - m2 - color - fill - size - linetype - alpha - lineend

Examples

```
dat <- data.frame(
  Group = c("A", "A", "B", "B", "C"),
  Type = c("X", "Y", "X", "Y", "X")
)
long <- make_long(dat, Group, Type)
ggplot2::ggplot(
  long,
  ggplot2::aes(
    x = x,
    next_x = next_x,
    node = node,
    next_node = next_node,
    fill = node
  )
) +
  geom_sankey() +
  theme_sankey()
```

`geom_sankey_bump` *geom_sankey_bump*

Description

Creates a sankey bump plot which visualize flows between nodes. Each observation needs to have a ‘x‘ aesthetic as well as a ‘next_x‘ column which declares where that observation should flow. Also, each observation should have a ‘node‘ and a ‘next_node‘ aesthetic which provide information about which group in the y-direction.

Usage

```
geom_sankey_bump(
  mapping = NULL,
  data = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  smooth = 8,
  type = "sankey",
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Provide you own mapping. Both x and y need to be numeric.
data	Provide you own data.
position	Change position.
na.rm	Remove missing values.
show.legend	Show legend in plot.
smooth	How much smooth should the curve have? More means steeper curve.
type	Either ‘sankey‘ or ‘alluvial‘.
inherit.aes	Should the geom inherit aesthetics.
...	Other arguments to be passed to the geom.

Details

Other important arguments are; ‘space‘ which provides the space between nodes in the y-direction; ‘shift‘ which shifts nodes in the y-direction.

Value

A ggplot layer.

---

geom_sankey_label *geom_sankey_label*

Description

Creates centered labels or text in nodes of your sankey plot. Needs to have the exact same aesthetics as the call to ‘geom_sankey‘ to work.

Usage

```
geom_sankey_label(
  mapping = NULL,
  data = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  space = NULL,
  type = "sankey",
  width = 0.1,
  inherit.aes = TRUE,
  ...
)

geom_sankey_text(
```

```
  mapping = NULL,
  data = NULL,
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  space = NULL,
  type = "sankey",
  width = 0.1,
  inherit.aes = TRUE,
  ...
)
```

Arguments

mapping	Provide you own mapping. Both x and y need to be numeric.
data	Provide you own data.
position	Change position.
na.rm	Remove missing values.
show.legend	Show legend in plot.
space	Space between nodes in the y-direction.
type	Either ‘sankey‘ or ‘alluvial‘.
width	Width of nodes.
inherit.aes	Should the geom inherit aesthetics.
...	Other arguments to be passed to the geom.

Value

A ggplot layer.

---

get_chinese_palettes *Get Chinese color palettes*

Description

Get Chinese color palettes

Usage

```
get_chinese_palettes(prefix = "Chinese")
```

Arguments

prefix	The prefix of the palette names. Default is ‘"Chinese_"‘.

Value

A list of Chinese color palettes.

Examples

```
show_palettes(get_chinese_palettes())
```

---

get_colors *Get colors from Chinese colors dataset or palettes*

Description

Search for colors in the Chinese colors dataset and all available palettes. This function can search by palette names, color names (pinyin or Chinese), numbers, or hex codes. It automatically searches in all palettes and reports which palette(s) contain the found colors.

Usage

```
get_colors(..., palettes = NULL)
```

Arguments

`...`	One or more search values. Can be palette names, color names (pinyin or Chinese), numbers, or hex codes. If NULL, using all Chinese colors. Can also be `ggplot`, `patchwork`, or other renderable plot objects (`gtable`, `grob`, `gList`, `ComplexHeatmap::Heatmap`, `pheatmap`, `trellis`) to get the colors used in the plots.
`palettes`	Optional. A named list of palettes to search in. If `NULL` (default), searches in all available palettes.

Value

A data frame with class `colors` containing matching color information. The result is automatically printed using `print.colors()`. For `ggplot` or `patchwork` objects, returns a two-column `colors` data frame with the factor levels (`factor`) and their mapped colors (`color`). For other renderable plot objects, returns a two-column `colors` data frame with the colors used in the plot (`color`) and their Chinese names (`name`).

See Also

chinese_colors for the dataset of Chinese traditional colors. get_chinese_palettes for getting Chinese color palettes. ChineseColors for the ChineseColors object.

Examples

```
get_colors("Paired")

get_colors("#FF7F00")

get_colors("pinlan")
get_colors(44)
get_colors("#2B73AF")

get_colors("cyan", palettes = "ChineseSet64")

# Get the factor-color mapping used in a ggplot object
library(ggplot2)
p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point() +
  scale_color_manual(values = get_colors("Paired")$hex)
get_colors(p)
```

`get_legend` *Extract legend from a plot*

Description

Extract the legend grob from a plot object. ggplot2 3.5+ places legends in `guide-box-left` / `right` / `top` / `bottom` / `inside` slots; unused slots are empty `zeroGrob`s. This helper returns the first non-empty legend, or the legend at `position` when that argument is set.

Usage

```
get_legend(plot, position = NULL)
```

Arguments

`plot`	A plot object.
`position`	Optional legend position to extract. One of `"left"`, `"right"`, `"top"`, `"bottom"`, or `"inside"`. Default is `NULL`, which returns the first non-empty legend grob.

Value

The legend grob, or `NULL` if the plot has no legend.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(wt, mpg, colour = factor(cyl))) +
  geom_point()
get_legend(p)
get_legend(p + theme(legend.position = "bottom"), position = "bottom")
```

get_vars *Get used vars in a ggplot object*

Description

Get used vars in a ggplot object

Usage

```
get_vars(p, reverse = FALSE, verbose = TRUE)
```

Arguments

p	A ggplot object.
reverse	Whether to return unused vars. Default is FALSE.
verbose	Whether to print the message. Default is TRUE.

Value

A character vector of variable names. If reverse is FALSE, returns used variables; if TRUE, returns unused variables.

Examples

```
library(ggplot2)
p <- ggplot(
  data = mtcars,
  aes(x = mpg, y = wt, colour = cyl)
) +
  geom_point()
get_vars(p)
get_vars(p, reverse = TRUE)
```

GraphPlot *Graph Plot*

Description

A function to plot a graph with nodes and edges.

Usage

```
GraphPlot(
  node,
  edge,
  transition = NULL,
  node_coord = c("x", "y"),
  node_group = NULL,
  node_palette = "Chinese",
  node_palcolor = NULL,
  node_size = 4,
  node_alpha = 1,
  node_highlight = NULL,
  node_highlight_color = "red",
  label = FALSE,
  label.size = 3.5,
  label.fg = "white",
  label.bg = "black",
  label.bg.r = 0.1,
  label_insitu = FALSE,
  label_repel = FALSE,
  label_repulsion = 20,
  label_point_size = 1,
  label_point_color = "black",
  label_segment_color = "black",
  edge_threshold = 0.01,
  use_triangular = c("upper", "lower", "both"),
  edge_line = c("straight", "curved"),
  edge_line_curvature = 0.3,
  edge_line_angle = 90,
  edge_color = "grey30",
  edge_size = c(0.2, 1),
  edge_alpha = 0.5,
  edge_shorten = 0,
  edge_offset = 0,
  edge_highlight = NULL,
  edge_highlight_color = "red",
  transition_threshold = 0.01,
  transition_line = c("straight", "curved"),
  transition_line_curvature = 0.3,
  transition_line_angle = 90,
  transition_color = "black",
  transition_size = c(0.2, 1),
  transition_alpha = 1,
  transition_arrow_type = "closed",
  transition_arrow_angle = 20,
  transition_arrow_length = grid::unit(0.02, "npc"),
  transition_shorten = 0.05,
  transition_offset = 0,
```

```
  transition_highlight = NULL,
  transition_highlight_color = "red",
  aspect.ratio = 1,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
  ylab = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  theme_use = "theme_this",
  theme_args = list(),
  return_layer = FALSE
)
```

Arguments

node	A data frame representing the nodes of the graph.
edge	A matrix representing the edges of the graph.
transition	A matrix representing the transitions between nodes.
node_coord	A character vector specifying the names of the columns in node that represent the x and y coordinates.
node_group	A character vector specifying the name of the column in node that represents the grouping of the nodes.
node_palette	A character vector specifying the name of the color palette for node groups.
node_palcolor	A character vector specifying the names of the colors for each node group.
node_size	A numeric value or column name of node specifying the size of the nodes.
node_alpha	A numeric value or column name of node specifying the transparency of the nodes.
node_highlight	A character vector specifying the names of nodes to highlight.
node_highlight_color	A character vector specifying the color for highlighting nodes.
label	Whether to show labels for the nodes.
label.size	The size of the labels.
label.fg	A character vector specifying the foreground color of the labels.
label.bg	A character vector specifying the background color of the labels.
label.bg.r	The background color transparency of the labels.
label_insitu	Whether to display the node group labels in situ or as numeric values.
label_repel	Whether to use force-directed label repulsion.
label_repulsion	The repulsion force for labels.
label_point_size	The size of the label points.

label_point_color
: A character vector specifying the color of the label points.

label_segment_color
: A character vector specifying the color for the label segments.

edge_threshold
: The threshold for removing edges.

use_triangular
: A character vector specifying which part of the edge matrix to use (upper, lower, both).

edge_line
: A character vector specifying the type of line for edges (straight, curved).

edge_line_curvature
: The curvature of curved edges.

edge_line_angle
: The angle of curved edges.

edge_color
: A character vector specifying the color of the edges.

edge_size
: A numeric vector specifying the range of edge sizes.

edge_alpha
: The transparency of the edges.

edge_shorten
: The length of the edge shorten.

edge_offset
: The length of the edge offset.

edge_highlight
: A character vector specifying the names of edges to highlight.

edge_highlight_color
: A character vector specifying the color for highlighting edges.

transition_threshold
: The threshold for removing transitions.

transition_line
: A character vector specifying the type of line for transitions (straight, curved).

transition_line_curvature
: The curvature of curved transitions.

transition_line_angle
: The angle of curved transitions.

transition_color
: A character vector specifying the color of the transitions.

transition_size
: A numeric vector specifying the range of transition sizes.

transition_alpha
: The transparency of the transitions.

transition_arrow_type
: A character vector specifying the type of arrow for transitions (closed, open).

transition_arrow_angle
: The angle of the transition arrow.

transition_arrow_length
: The length of the transition arrow.

transition_shorten
: The length of the transition shorten.

`transition_offset`
: The length of the transition offset.

`transition_highlight`
: A character vector specifying the names of transitions to highlight.

`transition_highlight_color`
: A character vector specifying the color for highlighting transitions.

`aspect.ratio`
: Aspect ratio passed to `theme()`.

`title`
: Plot title.

`subtitle`
: Plot subtitle.

`xlab`
: Label for the x axis.

`ylab`
: Label for the y axis.

`legend.position`
: Legend position passed to `theme()`.

`legend.direction`
: Legend direction passed to `theme()`.

`theme_use`
: Theme function applied to the plot.

`theme_args`
: A named list of arguments passed to `theme_use`.

`return_layer`
: Whether to return the plot layers as a list. Defaults is `FALSE`.

Examples

```
node <- data.frame(
  x = c(0.2, 0.3, 0.5, 0.3),
  y = c(0.2, 0, 0.6, 0.7),
  group = c("A", "B", "C", "B"),
  row.names = c("n1", "n2", "n3", "n4")
)
edge <- matrix(
  c(
    0, 0.4, 0.2, 0.1,
    0.4, 0, 0.3, 0.2,
    0.2, 0.3, 0, 0.5,
    0.2, 0.2, 0.5, 0.3
  ),
  nrow = 4,
  byrow = TRUE,
  dimnames = list(rownames(node), rownames(node))
)
GraphPlot(
  node = node,
  edge = edge
)

GraphPlot(
  node = node,
  edge = edge,
  node_group = "group",
  node_highlight = "n1",
```

```
  label = TRUE,
  label_repel = TRUE
)
```

`grid_draw` *Draw grobs at specified positions*

Description

Draw a list of grobs at specified positions with given widths and heights.

Usage

```
grid_draw(groblist, x, y, width, height)
```

Arguments

`groblist`	A grob or a list of grobs to draw.
`x`	A numeric vector of x positions for each grob.
`y`	A numeric vector of y positions for each grob.
`width`	A numeric vector of widths for each grob.
`height`	A numeric vector of heights for each grob.

Value

No return value, called for side effects (drawing grobs).

`head.colors` *Return the first part of a colors object*

Description

Returns the first part of a colors object, similar to `head()` for data frames.

Usage

```
## S3 method for class 'colors'
head(x, n = 6L, ...)
```

Arguments

`x`	A colors object (data frame with color information).
`n`	Number of rows to return. Default is `6`.
`...`	Additional arguments passed to `head()`.

Value

A colors object with the first n rows.

Examples

```
head(get_colors())

head(get_colors(), n = 10)
```

`heatmap_border_spec` *Normalize ComplexHeatmap border settings*

Description

Normalize a heatmap border flag, optional colour override, and line width into values accepted by ‘ComplexHeatmap::Heatmap‘ and ‘ComplexHeatmap::Legend‘.

Usage

```
heatmap_border_spec(border = NULL, palcolor = "black", size = 1)

heatmap_border_enabled(border)

heatmap_border_color(border, palcolor = "black")

heatmap_border_size_value(size)

heatmap_border_gp(border, color, size = 1)

heatmap_legend_border(border, color)
```

Arguments

`border`	‘NULL‘, a logical border flag, or a non-empty colour string.
`palcolor`	Fallback border colour when ‘border‘ is not a colour string.
`size`	Border line width. Invalid or negative values use ‘1‘.
`color`	Fallback border colour used by the ‘gp‘ and legend helpers.

Value

A list with ‘enabled‘, ‘color‘, ‘size‘, ‘gp‘, and ‘legend_border‘.

heatmap_fixsize *Compute fixed heatmap device size*

Description

Compute fixed heatmap device size

Usage

```
heatmap_fixsize(
  width,
  width_sum,
  height,
  height_sum,
  units,
  ht_list,
  legend_list
)
```

Arguments

width	Optional user-provided target width. If NULL, width is estimated from the drawn heatmap layout.
width_sum	Numeric baseline width used when heatmap body size is in npc units.
height	Optional user-provided target height. If NULL, height is estimated from the drawn heatmap layout.
height_sum	Numeric baseline height used when heatmap body size is in npc units.
units	Output unit string passed to grid conversion helpers, such as "in", "cm", or "mm".
ht_list	A ComplexHeatmap::Heatmap or ComplexHeatmap::HeatmapList object.
legend_list	A list of ComplexHeatmap::Legend objects.

Value

A list with two converted units: ht_width and ht_height.

`heatmap_rendersize` *Estimate heatmap render size*

Description

Estimate heatmap render size

Usage

```
heatmap_rendersize(
  width,
  height,
  units,
  ha_top_list,
  ha_left,
  ha_right,
  ht_list,
  legend_list,
  flip
)
```

Arguments

`width`	Numeric vector of heatmap body widths. Interpretation depends on `flip`.
`height`	Numeric vector of heatmap body heights. Interpretation depends on `flip`.
`units`	Unit string for numeric-to-grid conversion (for example, `"in"`, `"cm"`, `"mm"`).
`ha_top_list`	A list of top annotations (typically `ComplexHeatmap::HeatmapAnnotation` objects).
`ha_left`	Optional left annotation.
`ha_right`	Optional right annotation.
`ht_list`	A `ComplexHeatmap::Heatmap` or `ComplexHeatmap::HeatmapList` object.
`legend_list`	A list containing legend objects; `NULL` entries are ignored.
`flip`	Logical; when `TRUE`, width/height interpretation follows the flipped layout branch.

Value

A list with numeric `width_sum` and `height_sum` in `units`.

`invert_figures` *Invert the colours of a set of figures*

Description

Build dark-mode variants of several figures at once, dispatching on the file extension: SVG files are inverted with `invert_svg()` and PNG files with `invert_png()`. Every output is written next to its input, or into `output_dir`, with a `-dark` suffix added to the name. Files that already carry the suffix are skipped, so running the function twice does not complement the dark variants again.

Usage

```
invert_figures(
  path,
  output_dir = NULL,
  suffix = "-dark",
  recursive = FALSE,
  quiet = FALSE
)
```

Arguments

`path`	Path(s) to figure files, or to directories holding them.
`output_dir`	Directory the dark variants are written to. Default is next to each input file.
`suffix`	Suffix added to the file name of every dark variant. Default is `-dark`.
`recursive`	Whether a directory is scanned recursively. Default is `FALSE`.
`quiet`	Whether to suppress the per-file messages. Default is `FALSE`.

Details

Other formats are not supported: there is no way to rewrite the colours of a PDF from R without rasterising it first. Re-render the figure as SVG or PNG where possible, for instance with `ggsave()`, or rasterise the PDF before inverting it. Unsupported files are only reported when they are passed as a file; a directory is scanned for the supported extensions.

Value

The paths of the written files, invisibly.

invert_png *Invert the colours of a PNG file*

Description

Build a dark-mode variant of a PNG image by replacing every colour channel with its RGB complement. The alpha channel is kept as it is, so a transparent background stays transparent.

Usage

```
invert_png(path, output = NULL, quiet = FALSE)
```

Arguments

path	Path to the input PNG file.
output	Path of the output PNG. Defaults to the input name with a -dark suffix next to it.
quiet	Whether to suppress the message about the written file. Default is FALSE.

Value

The output path, invisibly.

invert_svg *Invert the colours of an SVG file*

Description

Build a dark-mode variant of an SVG image by replacing every colour with its RGB complement. Both the hexadecimal notation used by svglite and the rgb() notation written by cairo-based devices such as grDevices::svg() are understood. Elements that do not set their own fill would keep the SVG default black, so default_fill is written on the root <svg> element, where those elements inherit it.

Usage

```
invert_svg(
  path,
  output = NULL,
  default_fill = "#ffffff",
  invert_raster = TRUE,
  quiet = FALSE
)
```

Arguments

path	Path to the input SVG file.
output	Path of the output SVG. Defaults to the input name with a -dark suffix next to it. Use NA to return the modified SVG as a string without writing a file.
default_fill	Fill written on the root <svg> element, so elements without their own fill do not stay black. Use NULL to leave the root element untouched.
invert_raster	Whether to complement the colours of embedded raster images as well. Default is TRUE. Requires the png package.
quiet	Whether to suppress the message about the written file. Default is FALSE.

Value

The output path, invisibly, or the modified SVG as a string when output is NA.

---

jitter_highlighted_points

Jitter highlighted points deterministically

Description

Jitter highlighted points deterministically

Usage

```
jitter_highlighted_points(
  data,
  jitter_width = 0.2,
  jitter_height = 0.2,
  seed = 11
)
```

Arguments

data	A data frame containing ‘x‘, ‘y‘, and ‘border‘ columns.
jitter_width	Width of the x jitter.
jitter_height	Height of the y jitter.
seed	Numeric value used to offset the deterministic sequence.

Value

A data frame with ‘x_plot‘ and ‘y_plot‘ columns.

Examples

```
df <- data.frame(
  x = c(0, 1),
  y = c(0, 1),
  border = c(FALSE, TRUE)
)
jitter_highlighted_points(df, jitter_width = 0.2, jitter_height = 0.2)
```

---

major_grid *Set major grid lines on a ggplot*

Description

Add or remove major grid lines from a ggplot object.

Usage

```
major_grid(
  plot,
  grid_major = TRUE,
  grid_major_colour = "grey80",
  grid_major_linetype = 2,
  grid_major_linewidth = 0.3
)
```

Arguments

plot	A ggplot object.
grid_major	Whether to show major grid lines. Default is TRUE.
grid_major_colour	Grid line colour. Default is "grey80".
grid_major_linetype	Grid line type. Default is 2 (dashed).
grid_major_linewidth	Grid line width. Default is 0.3.

Value

A ggplot object with modified panel.grid.major theme element.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(wt, mpg)) +
  geom_point()
major_grid(p, grid_major = TRUE)
```

make_long *Make a long data frame for sankey plot*

Description

Prepares a 'wide' data frame into a format that 'geom_sankey' or 'geom_alluvial' understands. Useful to show flows between dimensions in dataset.

Usage

```
make_long(.df, ..., value = NULL)
```

Arguments

.df	a data frame
...	unquoted columnnames of df that you want to include in the plot.
value	if each row have a weight this weight could be kept by providing column name of weight.

Value

a longer data frame

mestimate *Estimate the fuzzifier parameter m*

Description

Estimate the fuzzifier parameter m for fuzzy clustering based on the data dimensions.

Usage

```
mestimate(data)
```

Arguments

data	A matrix or data frame.

Value

The estimated fuzzifier parameter m.

NetworkPlot *Network Plot*

Description

Network Plot

Usage

```
NetworkPlot(
  edge,
  node = NULL,
  from = "from",
  to = "to",
  weight = "weight",
  node_name = NULL,
  edge_group = NULL,
  edge_palcolor = NULL,
  edge_palette = "Chinese",
  node_group = NULL,
  node_palcolor = NULL,
  node_palette = "Chinese",
  node_shape = NULL,
  node_shape_values = NULL,
  node_size = NULL,
  node_size_range = NULL,
  layout = c("fr", "kk", "circle", "none"),
  node_coord = c("x", "y"),
  edge_width = c(0.2, 1),
  edge_width_legend = FALSE,
  edge_alpha = 0.7,
  edge_curvature = 0,
  edge_arrow = FALSE,
  label = TRUE,
  label_nodes = NULL,
  label.size = 3.5,
  label_size = NULL,
  label_face = NULL,
  label.fg = "black",
  label.bg = "white",
  label.bg.r = 0.1,
  highlight = NULL,
  highlight.color = "red",
  highlight.size = 6,
  title = NULL,
  aspect.ratio = 1,
  legend.position = "right",
```

```
  theme_use = "theme_this",
  theme_args = list(),
  seed = 42
)
```

Arguments

edge	A data frame with one row per edge.
node	Optional data frame with one row per node. Node identifiers are taken from node_name, from a column named name, or from the row names.
from, to	Column names in edge holding the edge endpoints.
weight	Column name in edge holding edge weights. Use NULL for unweighted edges.
node_name	Column name in node holding node identifiers.
edge_group	Optional column name in edge used to colour edges.
edge_palcolor, edge_palette	Custom colors or palette name for edge_group.
node_group	Optional column name in node used to fill nodes.
node_palcolor, node_palette	Custom colors or palette name for node_group.
node_shape	Optional node shapes. Either a column name in node or a named vector of grid shapes keyed by node name.
node_shape_values	Named vector mapping node_shape levels to shapes.
node_size	Optional node sizes. Either a column name in node or a numeric value used for every node.
node_size_range	Optional size range used to rescale a numeric column supplied to node_size; NULL keeps the supplied values.
layout	Layout used when node has no coordinates. One of "fr", "kk", "circle" or "none".
node_coord	Column names in node holding x and y coordinates. Used when layout = "none".
edge_width	Width range for the edges; NULL keeps the ggplot2 default width scale.
edge_width_legend	Whether to show the edge width legend. Default is FALSE.
edge_alpha	Edge alpha.
edge_curvature	Edge curvature passed to geom_curve; 0 draws straight segments.
edge_arrow	Whether to draw arrow heads on edges.
label	Whether to draw node labels.
label_nodes	Nodes to label. Default labels every node.
label.size	Label text size.
label_size	Optional per-node label sizes. Either a column name in node, a named vector keyed by node name, or NULL to use label.size.

label_face Optional per-node label faces. Either a column name in node, a named vector keyed by node name, or NULL for plain text.

label.fg, label.bg, label.bg.r
Label foreground color, outline color and outline radius.

highlight Optional node names drawn with an extra ring.

highlight.color, highlight.size
Color and size of the highlight ring.

title Optional plot title.

aspect.ratio Aspect ratio passed to the theme.

legend.position
Legend position passed to the theme.

theme_use Theme function name, one of "theme_this", "theme_blank" or "theme_void".

theme_args A named list of arguments passed to theme_use.

seed Optional integer seed for the layout.

Value

A ggplot object.

Examples

```
edge <- data.frame(
  from = c("A", "A", "B", "C"),
  to = c("B", "C", "C", "D"),
  weight = c(1, -2, 3, 1)
)
NetworkPlot(edge)

node <- data.frame(
  name = c("A", "B", "C", "D"),
  group = c("g1", "g2", "g2", "g3"),
  shape = c("hub", "leaf", "leaf", "leaf")
)
NetworkPlot(
  edge,
  node = node,
  node_group = "group",
  node_shape = "shape",
  node_shape_values = c("hub" = 23, "leaf" = 21),
  label_nodes = c("A", "B")
)
```

normalize_drawable *Normalize drawable objects to grobs*

Description

Normalize drawable objects to grobs

Usage

```
normalize_drawable(obj)
```

Arguments

obj	A drawable object, such as a ggplot, patchwork, or grid grob.

Value

A grid grob object. If the input is not drawable, returns a null grob.

Examples

```
library(ggplot2)
p <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
  ggplot2::geom_point()
g <- normalize_drawable(p)
p + g
```

no_legend *Remove the legend*

Description

Remove the legend

Usage

```
no_legend()
```

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg, colour = factor(cyl))) +
  geom_point() +
  no_legend()
```

no_margin *Remove plot margins*

Description

Remove plot margins

Usage

```
no_margin()
```

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  no_margin()
```

no_x_text *Remove x-axis text and ticks*

Description

Remove x-axis text and ticks

Usage

```
no_x_text()
```

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  no_x_text()
```

no_y_text *Remove y-axis text and ticks*

Description

Remove y-axis text and ticks

Usage

```
no_y_text()
```

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  no_y_text()
```

palette_colors *Color palettes collected*

Description

This function creates a color palette for a given vector of values.

Usage

```
palette_colors(
  x,
  n = 100,
  palette = "Chinese",
  palcolor = NULL,
  type = c("auto", "discrete", "continuous"),
  matched = FALSE,
  reverse = FALSE,
  NA_keep = FALSE,
  NA_color = "grey80"
)
```

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.
matched	Whether to return a color vector of the same length as x. Default is FALSE.
reverse	Whether to invert the colors. Default is FALSE.
NA_keep	Whether to keep the color assignment to NA in x. Default is FALSE.
NA_color	Color assigned to NA if NA_keep is TRUE. Default is "grey80".

Value

A character vector of color codes (hexadecimal format) corresponding to the input values x. The length and structure depend on the matched parameter.

See Also

show_palettes, palette_list

Examples

```
x <- c(1:3, NA, 3:5)
(pal1 <- palette_colors(
  x,
  palette = "Spectral"
))
(pal2 <- palette_colors(
  x,
  palcolor = c("red", "white", "blue")
))
(pal3 <- palette_colors(
  x,
  palette = "Spectral",
  n = 10
))
(pal4 <- palette_colors(
  x,
  palette = "Spectral",
  n = 10,
  reverse = TRUE
))
(pal5 <- palette_colors(
  x,
  palette = "Spectral",
  matched = TRUE
```

```
))
(pal6 <- palette_colors(
  x,
  palette = "Spectral",
  matched = TRUE,
  NA_keep = TRUE
))
show_palettes(
  list(pal1, pal2, pal3, pal4, pal5, pal6)
)

# Use Chinese color palettes
palette_colors(
  x = letters[1:5],
  palette = "ChineseRed",
  type = "discrete"
)
palette_colors(
  x = letters[1:5],
  palette = "Chinese",
  type = "discrete"
)

all_palettes <- show_palettes(return_palettes = TRUE)
names(all_palettes)
```

`palette_list` *A list of palettes for use in data visualization*

Description

A list of palettes for use in data visualization

`panel_fix` *Set the panel width/height of a plot to a fixed value*

Description

The ggplot object, when stored, can only specify the height and width of the entire plot, not the panel. The latter is obviously more important to control the final result of a plot. This function can set the panel width/height of plot to a fixed value and rasterize it.

Usage

```
panel_fix(
  x = NULL,
  panel_index = NULL,
  respect = NULL,
  width = NULL,
  height = NULL,
  margin = 1,
  padding = 0,
  units = "in",
  raster = FALSE,
  dpi = 300,
  return_grob = FALSE,
  bg_color = "white",
  save = NULL,
  verbose = FALSE,
  ...
)

panel_fix_overall(
  x,
  panel_index = NULL,
  respect = NULL,
  width = NULL,
  height = NULL,
  margin = 1,
  units = "in",
  raster = FALSE,
  dpi = 300,
  return_grob = FALSE,
  bg_color = "white",
  save = NULL,
  verbose = TRUE
)
```

Arguments

x	A ggplot object, a grob object, or a combined plot made by patchwork or cowplot package.
panel_index	Specify the panel to be fixed. If NULL, will fix all panels.
respect	Whether row heights and column widths should respect each other.
width	The desired width of the fixed panels.
height	The desired height of the fixed panels.
margin	The margin to add around each panel, in inches. Default is 1.
padding	The padding to add around each panel, in inches. Default is 0.
units	The units in which height, width and margin are given. Can be "mm", "cm", "in", etc. See grid::unit.

raster	Whether to rasterize the panel.
dpi	Plot resolution.
return_grob	Whether to return a grob object instead of a wrapped patchwork object. Default is FALSE.
bg_color	The background color of the plot.
save	NULL or the file name used to save the plot.
verbose	Whether to print the message. Default is TRUE.
...	Additional arguments passed to other functions.

Value

If return_grob is TRUE, returns a gtable object. Otherwise, returns a patchwork object with fixed panel sizes. The returned object has a size attribute containing width, height, and units.

Examples

```
library(ggplot2)
p <- ggplot(
  data = mtcars, aes(x = mpg, y = wt, colour = cyl)
) +
  geom_point() +
  facet_wrap(~gear, nrow = 2)
# fix the size of panel
panel_fix(
  p,
  width = 5,
  height = 3,
  units = "cm"
)
# rasterize the panel
panel_fix(
  p,
  width = 5,
  height = 3,
  units = "cm",
  raster = TRUE,
  dpi = 90
)

# `panel_fix` will build and render the plot when input a ggplot object.
# so after `panel_fix`, the size of the object will be changed.
object.size(p)
object.size(
  panel_fix(
    p,
    width = 5,
    height = 3,
    units = "cm"
  )
)
```

patchwork_grob *Convert a patchwork object to a grob*

Description

Convert a patchwork object to a gtable grob by processing annotations and building the patchwork layout.

Usage

```
patchwork_grob(x, ...)
```

Arguments

x	A patchwork object.
...	Additional arguments passed to other functions.

Value

A gtable object.

plot_autocompose *Compose plots at their preferred sizes*

Description

Calculate each plot's preferred physical size and keep every plot at that size. The default publication layout searches compact asymmetric placements with aligned edges, while the ordered layout packs shorter rows continuously and centers them. Apply plot-specific styling before composing.

Usage

```
plot_autocompose(
  plots,
  ncol = NULL,
  nrow = NULL,
  panel_width = 45,
  panel_height = 35,
  margin = 2,
  units = "mm",
  save = NULL,
  dpi = 300,
  base_size = "auto",
  label_autosize = TRUE,
  target_aspect = 1.6,
```

```
  target_width = NULL,
  layout = c("publication", "rows"),
  gutter = 4
)
```

Arguments

plots	A plot or list of plots accepted by plot_autosize().
ncol, nrow	Number of columns and rows. Supplying either uses the ordered row layout.
panel_width, panel_height	Base panel width and height.
margin	Outer margin.
units	Units for panel dimensions and margin. See grid::unit().
save	NULL or a file name to save the fitted plot.
dpi	Resolution used when saving raster output.
base_size	"auto" to harmonize ggplot and grid text to one inferred base size in points, a positive point size, or NULL to preserve each component's typography.
label_autosize	Whether to enlarge constant geom_text, geom_label, and ggrepel label sizes when the panel has enough horizontal space.
target_aspect	Preferred overall width-to-height ratio when ncol and nrow are both NULL.
target_width	Optional publication target width in units. Limits canvas width to fit journal single-column (e.g. 85 mm) or double-column (e.g. 180 mm) constraints.
layout	"publication" for compact asymmetric placement or "rows" to preserve input order in rows.
gutter	Space between components in units for the publication layout.

Value

A wrapped plot with size and autosize attributes. Component sizes are stored in attr(x, "autosize")$components.

Examples

```
p1 <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
  ggplot2::geom_point()
p2 <- p1 + ggplot2::labs(title = "A wider plot title")
plot_autocompose(list(p1, p2), ncol = 2)
```

`plot_autosize` *Automatically size a plot*

Description

Build a compact plot by choosing a physical panel size and expanding it for dense in-panel text or overflowing plot labels. A shared point-size baseline harmonizes ggplot, patchwork, and recorded grid text such as wrapped ComplexHeatmap output while preserving their relative hierarchy. Constant data-label text is enlarged conservatively when space permits. The calculation delegates final layout and saving to `panel_fix()` while preserving label content, fixed aspect ratios, patchwork layouts, and that function's behaviour.

Usage

```
plot_autosize(
  x,
  panel_width = 45,
  panel_height = 35,
  margin = 2,
  units = "mm",
  save = NULL,
  dpi = 300,
  base_size = "auto",
  label_autosize = TRUE,
  verbose = FALSE
)
```

Arguments

`x`	A ggplot object, grob, or combined plot accepted by `panel_fix()`.
`panel_width, panel_height`	Base panel width and height.
`margin`	Outer margin.
`units`	Units for panel dimensions and margin. See `grid::unit()`.
`save`	`NULL` or a file name to save the fitted plot.
`dpi`	Resolution used when saving raster output.
`base_size`	`"auto"` to harmonize ggplot and grid text to one inferred base size in points, a positive point size, or `NULL` to preserve each component's typography.
`label_autosize`	Whether to enlarge constant `geom_text`, `geom_label`, and ggrepel label sizes when the panel has enough horizontal space.
`verbose`	Whether to report messages from `panel_fix()`.

Value

A plot with `size` and `autosize` attributes. Ordinary plots are returned through `panel_fix()`; wrapped grid content is preserved directly.

Examples

```
library(ggplot2)
p <- ggplot(mtcars, aes(wt, mpg, colour = factor(cyl))) +
  geom_point()
p1 <- p +
  labs(title = "Fuel economy by vehicle weight")
p1

p2 <- plot_autosize(p)
p2

attr(p2, "size")

plot_autosize(p1)
```

---

plot_histogram *Histogram plot*

Description

Plot a numeric vector as a histogram with `theme_this` and thisplot palettes.

Usage

```
plot_histogram(
  data,
  binwidth = 0.01,
  show_border = FALSE,
  border_color = "black",
  alpha = 1,
  palette = "viridis",
  palcolor = NULL,
  xlab = "Value",
  ylab = "Count",
  legend.position = "right",
  theme_use = "theme_this",
  theme_args = list()
)
```

Arguments

`data`	A numeric vector.
`binwidth`	Width of the bins.
`show_border`	Whether to show bin borders.
`border_color`	Color of the bin borders.
`alpha`	Transparency of the bins.

palette	Palette name used to fill bins by count. See show_palettes.
palcolor	Custom colors used to create a color palette.
xlab, ylab	Axis titles.
legend.position	Legend position.
theme_use	Theme function applied to the plot.
theme_args	A named list of arguments passed to theme_use.

Value

A ggplot object.

Examples

```
set.seed(1)
plot_histogram(rnorm(500), binwidth = 0.2)
```

---

plot_scatter *Scatter plot*

Description

Plot two- or three-column data as a scatter plot with theme_this and thisplot palettes. Optional smoothing, correlation annotation, point density, and marginal plots are available.

Usage

```
plot_scatter(
  data,
  smoothing_method = c("lm", "loess"),
  palette = "Chinese",
  palcolor = NULL,
  title = NULL,
  xlab = NULL,
  ylab = NULL,
  legend.title = NULL,
  legend.position = "right",
  theme_use = "theme_this",
  theme_args = list(),
  margins = c("both", "x", "y"),
  marginal_type = NULL,
  margins_size = 10,
  compute_correlation = TRUE,
  compute_correlation_method = c("pearson", "spearman"),
  keep_aspect_ratio = TRUE,
  facet = FALSE,
```

```
  se = FALSE,
  pointdensity = TRUE
)
```

Arguments

`data`	A data frame with 2 columns (`x`, `y`) or 3 columns (`x`, `y`, and a grouping column). Extra columns are ignored.
`smoothing_method`	Smoothing method passed to ggplot2::geom_smooth. Can be `"lm"` or `"loess"`.
`palette`	Palette name used for groups or point-density coloring. See show_palettes.
`palcolor`	Custom colors used to create a color palette.
`title`, `xlab`, `ylab`	Plot titles.
`legend.title`	Legend title.
`legend.position`	Legend position.
`theme_use`	Theme function applied to the plot.
`theme_args`	A named list of arguments passed to `theme_use`.
`margins`, `marginal_type`, `margins_size`	Marginal plot controls used when `ggExtra` is installed. `marginal_type` can be `"density"`, `"histogram"`, `"boxplot"`, `"violin"`, or `"densigram"`.
`compute_correlation`	Whether to annotate Pearson or Spearman correlation.
`compute_correlation_method`	Correlation method: `"pearson"` or `"spearman"`.
`keep_aspect_ratio`	Whether to use a 1:1 aspect ratio.
`facet`	Whether to facet by the grouping column when `data` has 3 columns.
`se`	Whether to show smoothing uncertainty.
`pointdensity`	Whether to color ungrouped points by local density when `ggpointdensity` is installed.

Value

A ggplot object, or a `ggExtra` grob when marginal plots are added.

Examples

```
set.seed(1)
plot_scatter(data.frame(x = rnorm(80), y = rnorm(80)))
plot_scatter(
  data.frame(
    x = rnorm(80),
    y = rnorm(80),
    cluster = rep(c("A", "B"), each = 40)
  )
)
```

print.ChineseColors *Print* ChineseColors *object*

Description

Print ChineseColors object

Usage

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

Arguments

x	A ChineseColors object.
...	Additional arguments.

Value

Details of the ChineseColors object.

print.colors *Print colors object*

Description

Print colors object

Usage

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

Arguments

x	A colors object (data frame with color information).
...	Additional arguments passed to print.

Value

Details of the colors objec.

print.thisplot_logo *Print logo*

Description

Print logo

Usage

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

Arguments

x	Input information.
...	Other parameters.

Value

Print the ASCII logo

print_colored_table *Print a data frame as a color-highlighted aligned table*

Description

Prints a data frame as an aligned console table with ANSI colors. By default each column receives one distinct color from ‘palette‘; set ‘by = "row"‘ to use one color per row. Custom ‘color_by‘ specifications can still color individual values. Column widths are computed from visible ANSI text, so escape sequences and Chinese text do not break alignment in the console. Colors are emitted only when the output supports ANSI colors (a terminal or RStudio, or when ‘options(thisplot.colors = TRUE)‘ is set); captured output stays clean.

Usage

```
print_colored_table(
  x,
  color_by = NULL,
  colors = NULL,
  palette = "Chinese",
  by = "col",
  ...
)
```

Arguments

x	A data frame.
color_by	A character vector of column names to color, or a named list of color sources. If ‘NULL‘ (default), colors are assigned according to ‘by‘.
colors	Optional color source used for all ‘color_by‘ columns unless overridden by a named list in ‘color_by‘: a named hex vector whose names match the values of the colored columns, a plain hex vector recycled per row, or ‘NA‘ to fall back to ‘palette‘.
palette	Palette name passed to ‘palette_colors()‘ for default row or column colors, or columns without explicit colors.
by	Whether default palette colors are assigned by ‘"col"‘ (default) or ‘"row"‘.
...	Additional arguments passed to ‘format()‘.

Value

‘x‘ invisibly.

Examples

```
t1 <- data.frame(
  name = c("a", "b", "c"),
  col1 = c(1L, 2L, 3L),
  col2 = c(4L, 5L, 6L)
)

print_colored_table(t1)
```

---

rasterise_plot *Rasterise the layers of a plot*

Description

Rasterise the layers of a plot

Usage

```
rasterise_plot(
  plot,
  layers = c("Point", "Tile", "Path", "Line", "Segment"),
  dpi = 300,
  dev = "ragg",
  recurse = TRUE
)
```

Arguments

plot	A ggplot or patchwork object.
layers	Layer types to rasterise, named without the Geom prefix, so the default "Point" matches geom_point() layers. Add "Violin", "Boxplot" or "Crossbar" to rasterise those layers as well.
dpi	Resolution of the rasterised layers, in dots per inch. Default is 300.
dev	Graphic device used to render the rasterised layers, passed to ggrastr::rasterise(). Default is "ragg", which does not need X11; the "cairo" device of the underlying package does, and is unavailable on many headless machines.
recurse	Whether to rasterise the sub-plots of a patchwork composite. Default is TRUE.

Value

The plot object with its heavy layers rasterised. Objects that are neither a ggplot nor a patchwork are returned unchanged, with a warning.

See Also

invert_svg()

Examples

```
panel <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
  ggplot2::geom_point() +
  ggplot2::facet_wrap(~cyl)

if (requireNamespace("ggrastr", quietly = TRUE)) {
  rasterise_plot(panel)
  rasterise_plot(patchwork::wrap_plots(panel, panel), dpi = 150)
}
```

RGBA2RGB *Convert RGBA color to RGB with background*

Description

Convert an RGBA (Red, Green, Blue, Alpha) color to RGB by compositing it with a background color based on the alpha channel.

Usage

```
RGBA2RGB(RGBA, BackGround = c(1, 1, 1))
```

Arguments

RGBA	A list containing RGB values and alpha channel.
BackGround	The background RGB color to composite with. Default is c(1, 1, 1) (white).

Value

A numeric vector of RGB values.

`segements_df` *Shorten and offset the segment*

Description

This function takes a data frame representing segments in a plot and shortens and offsets them based on the provided arguments.

Usage

```
segements_df(data, shorten_start, shorten_end, offset)
```

Arguments

`data`	A data frame containing the segments. It should have columns 'x', 'y', 'xend', and 'yend' representing the start and end points of each segment.
`shorten_start`	The amount to shorten the start of each segment by.
`shorten_end`	The amount to shorten the end of each segment by.
`offset`	The amount to offset each segment by.

Value

The modified data frame with the shortened and offset segments.

Examples

```
library(ggplot2)
temp_nodes <- data.frame(
  "x" = c(10, 40),
  "y" = c(10, 30)
)
data <- data.frame(
  "x" = c(10, 40),
  "y" = c(10, 30),
  "xend" = c(40, 10),
  "yend" = c(30, 10)
)

ggplot(temp_nodes, aes(x = x, y = y)) +
  geom_point(size = 12) +
  xlim(0, 50) +
  ylim(0, 50) +
  geom_segment(
    data = data,
    aes(x = x, xend = xend, y = y, yend = yend)
```

```
  )

ggplot(temp_nodes, aes(x = x, y = y)) +
  geom_point(size = 12) +
  xlim(0, 50) +
  ylim(0, 50) +
  geom_segment(
    data = segements_df(
      data,
      shorten_start = 2,
      shorten_end = 3,
      offset = 1
    ),
    aes(x = x, xend = xend, y = y, yend = yend)
  )
```

`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. Default is `NULL`.
`type`	The type of palettes to include. Default is `"discrete"`.
`index`	The indices of the palettes to include. Default is `NULL`.
`palette_names`	The names of the palettes to include. Default is `NULL`.
`return_names`	Whether to return the names of the selected palettes. Default is `TRUE`.
`return_palettes`	Whether to return the colors of selected palettes. Default is `FALSE`.

Value

If `return_palettes` is `TRUE`, returns a list of color palettes. If `return_names` is `TRUE` (default), returns a character vector of palette names. Otherwise, returns `NULL` (called for side effects to display the plot).

See Also

palette_colors, 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", "Chinese"
  ),
  return_palettes = TRUE
)
# Include Chinese palettes via prefix
show_palettes(
  palette_names = c("ChineseRed", "ChineseBlue"),
  return_palettes = TRUE
)
```

---

simple_colors *Simple random color selection*

Description

Simple random color selection

Usage

```
simple_colors(n = 10, palette = NULL)
```

Arguments

n	The number of colors to return. Default is 10.
palette	The name of the palette to use. Default is NULL, colors will be selected from ChineseColors. Otherwise, colors will be selected from the specified palette. Available palette names can be queried with show_palettes.

Value

A character vector of hexadecimal color codes.

Examples

```
simple_colors()

show_palettes(simple_colors(n = 5))

# Get colors from a specific palette
simple_colors(n = 10, palette = "Paired")
simple_colors(n = 10, palette = "ChineseBlue")
simple_colors(n = 10, palette = "Spectral")
```

---

slim_data *Slim unused data in the plot*

Description

Remove unused columns from the data in a ggplot or patchwork object. This function keeps only the columns that are actually used in the plot (e.g., in mappings, aesthetics, or facets), which can significantly reduce the object size when the original data contains many unused columns.

Usage

```
slim_data(p)

## S3 method for class 'ggplot'
slim_data(p)

## S3 method for class 'patchwork'
slim_data(p)
```

Arguments

p	A ggplot object or a patchwork object.

Value

A ggplot or patchwork object with unused data columns removed.

Examples

```
library(ggplot2)
p <- ggplot(
  data = mtcars,
  aes(x = mpg, y = wt, colour = cyl)
) +
  geom_point()
object.size(p)
colnames(p$data)

p_slim <- slim_data(p)
object.size(p_slim)
colnames(p_slim$data)
```

`standardise` *Standardize data by rows*

Description

Standardize each row of a data matrix by subtracting the mean and dividing by the standard deviation.

Usage

```
standardise(data)
```

Arguments

`data`	A matrix or data frame to standardize.

Value

The standardized data with the same structure as input.

`StatPlot` *Statistic Plot*

Description

Visualizes data using various plot types such as bar plots, rose plots, ring plots, pie charts, trend plots, area plots, dot plots, sankey plots, chord plots, venn diagrams, and upset plots.

Usage

```
StatPlot(
  meta.data,
  stat.by,
  group.by = NULL,
  split.by = NULL,
  bg.by = NULL,
  flip = FALSE,
  NA_color = "grey",
  NA_stat = TRUE,
  keep_empty = FALSE,
  individual = FALSE,
  stat_level = NULL,
 plot_type = c("bar", "rose", "ring", "pie", "trend", "trend_alluvial", "area", "dot",
    "sankey", "chord", "venn", "upset"),
  venn_engine = c("ggVennDiagram", "venny"),
  venn_args = list(),
  stat_type = c("percent", "count", "value"),
  value.by = NULL,
  top_n = Inf,
  rank.by = c("absolute", "value"),
  complete_groups = NULL,
  value_cutoff = NULL,
  value_limits = NULL,
  value_midpoint = 0,
  value_legend_title = NULL,
  bar_fill = NULL,
  bar_width = 0.8,
  point_size = c(2.5, 7),
  character_width = 50,
  return_data = FALSE,
  position = c("stack", "dodge"),
  palette = "Chinese",
  palcolor = NULL,
  alpha = 1,
  bg_palette = "Chinese",
  bg_palcolor = NULL,
  bg_alpha = 0.2,
  label = FALSE,
  label_cutoff = NULL,
  label.size = 3.5,
  label.fg = "black",
  label.bg = "white",
  label.bg.r = 0.1,
  aspect.ratio = NULL,
  title = NULL,
  subtitle = NULL,
  xlab = NULL,
```

```
  ylab = NULL,
  legend.position = "right",
  legend.direction = "vertical",
  theme_use = "theme_this",
  theme_args = list(),
  x_text_angle = 45,
  grid_major = TRUE,
  grid_major_colour = "grey80",
  grid_major_linetype = 2,
  grid_major_linewidth = 0.3,
  combine = TRUE,
  nrow = NULL,
  ncol = NULL,
  byrow = TRUE,
  force = FALSE,
  seed = 11
)
```

Arguments

meta.data	The data frame containing the data to be plotted.
stat.by	The column name(s) in meta.data specifying the variable(s) to be plotted.
group.by	The column name(s) in meta.data specifying the grouping variable(s). Default is NULL.
split.by	The column name in meta.data specifying the variable to split plots by. Default is NULL.
bg.by	The column name in meta.data specifying the background variable for bar plots.
flip	Whether to flip the plot. Default is FALSE.
NA_color	The color to use for missing values.
NA_stat	Whether to include missing values in the plot. Default is TRUE.
keep_empty	Whether to keep empty groups in the plot. Default is FALSE.
individual	Whether to plot individual groups separately. Default is FALSE.
stat_level	The level(s) of the variable(s) specified in stat.by to include in the plot. Default is NULL.
plot_type	The type of plot to create. Can be one of "bar", "rose", "ring", "pie", "trend", "trend_alluvial", "area", "dot", "sankey", "chord", "venn", or "upset".
venn_engine	The engine used when plot_type = "venn". Can be one of "ggVennDiagram" or "venny". Default is "ggVennDiagram". The "venny" engine supports 2 to 4 sets.
venn_args	A list of additional arguments passed to venny::venny when plot_type = "venn" and venn_engine = "venny". The data argument is generated internally and cannot be supplied here.

stat_type	The type of statistic to compute for the plot. Can be one of "percent", "count", or "value". Continuous value mode supports plot_type = "bar" and plot_type = "dot".
value.by	Numeric column used when stat_type = "value".
top_n	Maximum number of records retained per group and split in value mode. Default is Inf.
rank.by	Rank continuous values by absolute magnitude or signed value.
complete_groups	For value dot plots, retain every available group for a label selected in any group within the same split.
value_cutoff	Optional minimum absolute value.
value_limits	Optional continuous colour limits. Symmetric limits are inferred by default.
value_midpoint	Midpoint of the continuous colour scale.
value_legend_title	Legend title for continuous values.
bar_fill	Optional fixed fill used by value bars.
bar_width	Width of bars in value mode.
point_size	Size range for value dot plots.
character_width	Maximum wrapped label width in value mode.
return_data	Return the plot and prepared data in value mode.
position	The position adjustment for the plot. Can be one of "stack" or "dodge".
palette	The name of the color palette to use. Default is "Chinese".
palcolor	Custom colors to use instead of palette. Default is NULL.
alpha	The transparency level for the plot.
bg_palette	The name of the background color palette to use for bar plots.
bg_palcolor	The color to use in the background color palette.
bg_alpha	The transparency level for the background color in bar plots.
label	Whether to add labels on the plot. Default is FALSE.
label_cutoff	Optional minimum plotted value required to show a label. Labels are shown only for values strictly greater than this cutoff. For percentage plots, use proportions; for example, label_cutoff = 0.1 shows labels only for values greater than 10%. Default is NULL, which labels all values.
label.size	The size of the labels.
label.fg	The foreground color of the labels.
label.bg	The background color of the labels.
label.bg.r	The radius of the rounded corners of the label background.
aspect.ratio	Aspect ratio of the panel. Default is NULL.
title	The title of the plot. Default is NULL.
subtitle	The subtitle of the plot. Default is NULL.

xlab	The label for the x-axis. Default is NULL.
ylab	The label for the y-axis. Default is NULL.
legend.position	The position of the legend. Can be one of "none", "left", "right", "bottom", "top", or a two-element numeric vector. Default is "right".
legend.direction	The direction of the legend. Can be one of "vertical" or "horizontal". Default is "vertical".
theme_use	The theme to use for the plot. Default is "theme_this".
theme_args	Additional arguments to pass to the theme function. Default is list().
x_text_angle	Rotation angle for x-axis labels. Default is 45. Must be a finite number.
grid_major	Whether to show major panel grid lines. Default is TRUE.
grid_major_colour	Color of major panel grid lines.
grid_major_linetype	Linetype of major panel grid lines.
grid_major_linewidth	Line width of major panel grid lines.
combine	Whether to combine multiple plots into one. Default is TRUE.
nrow	Number of rows when combining plots. Default is NULL.
ncol	Number of columns when combining plots. Default is NULL.
byrow	Whether to fill plots by row when combining. Default is TRUE.
force	Whether to force plotting even when variables have more than 100 levels. Default is FALSE.
seed	Random seed for reproducibility. Default is 11.

Examples

```
set.seed(1)
meta_data <- data.frame(
  Type = factor(
    sample(c("A", "B", "C"),
      50,
      replace = TRUE,
      prob = c(0.5, 0.3, 0.2)
    )
  ),
  Group = factor(sample(c("X", "Y", "Z"), 50, replace = TRUE)),
  Batch = factor(sample(c("B1", "B2"), 50, replace = TRUE))
)
meta_data$Region <- factor(
  ifelse(meta_data$Group %in% c("X", "Y"), "R1", "R2"),
  levels = c("R1", "R2")
)

StatPlot(
```

```
  meta_data,
  stat.by = "Type",
  group.by = "Group",
  split.by = "Batch",
  plot_type = "bar",
  position = "dodge"
)

StatPlot(
  meta_data,
  stat.by = "Type",
  group.by = "Group",
  stat_type = "count",
  plot_type = "ring",
  position = "dodge"
)

StatPlot(
  meta_data,
  stat.by = "Type",
  group.by = "Group",
  stat_type = "count"
)

StatPlot(
  meta_data,
  stat.by = "Type",
  plot_type = "pie"
)

StatPlot(
  meta_data,
  stat.by = "Type",
  group.by = "Group",
  stat_type = "count",
  plot_type = "area"
)

StatPlot(
  meta_data,
  stat.by = "Type",
  group.by = "Group",
  plot_type = "dot"
)

StatPlot(
  meta_data,
  stat.by = "Type",
  group.by = "Group",
  stat_type = "count",
  plot_type = "trend"
)
```

```
rank_data <- data.frame(
  Feature = paste0("Feature", 1:12),
  Group = rep(c("Control", "Treatment"), each = 6),
  Value = c(
    -2.4, 1.8, 1.2, -0.8, 0.6, 0.2,
    2.7, -2.1, 1.5, 0.9, -0.5, 0.3
  )
)

StatPlot(
  rank_data,
  stat.by = "Feature",
  value.by = "Value",
  group.by = "Group",
  stat_type = "value",
  plot_type = "bar",
  top_n = 4,
  flip = TRUE,
  palette = "RdBu"
)

StatPlot(
  rank_data,
  stat.by = "Feature",
  value.by = "Value",
  group.by = "Group",
  stat_type = "value",
  plot_type = "dot",
  top_n = 3,
  palette = "RdBu"
)

importance_data <- data.frame(
  Variable = paste0("Variable", 1:8),
  Importance = c(0.91, 0.78, 0.64, 0.52, 0.43, 0.31, 0.18, 0.09)
)
StatPlot(
  importance_data,
  stat.by = "Variable",
  value.by = "Importance",
  stat_type = "value",
  top_n = 5,
  flip = TRUE,
  bar_fill = "#3C8DBC"
)
```

---

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	The label of the x-axis.
ylab	The label of the y-axis.
lab_size	The size of the axis labels.
...	Arguments passed to the ggplot2::theme.

Value

A list containing ggplot2 theme objects and annotation objects. If add_coord is TRUE, returns a list with coordinate arrows; otherwise returns a list with theme only.

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_sankey *Themes for sankey plot*

Description

Minimal themes for sankey, alluvial and sankey bump plots

Usage

```
theme_sankey(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_alluvial(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)

theme_sankey_bump(
  base_size = 11,
  base_family = "",
  base_line_size = base_size/22,
  base_rect_size = base_size/22
)
```

Arguments

base_size	Base font size, given in pts.
base_family	Base font family.
base_line_size	Base size for line elements.
base_rect_size	Base size for rect elements.

theme_this *The default theme for scop plot function.*

Description

The default theme for scop plot function.

Usage

```
theme_this(aspect.ratio = NULL, base_size = 12, ...)
```

Arguments

aspect.ratio	Aspect ratio of the panel.
base_size	Base font size
...	Arguments passed to the ggplot2::theme.

Value

A ggplot2 theme object (class theme, gg).

Examples

```
library(ggplot2)
p <- ggplot(
  data = mtcars,
  aes(x = wt, y = mpg, colour = factor(cyl))
) +
  geom_point()
p + theme_this()
```

---

thisplot_logo *The logo of thisplot*

Description

The thisplot logo, using ASCII or Unicode characters Use cli::ansi_strip to get rid of the colors.

Usage

```
thisplot_logo(unicode = cli::is_utf8_output())
```

Arguments

unicode	Unicode symbols on UTF-8 platforms. Default is cli::is_utf8_output.

Value

A character vector with class thisplot_logo.

References

https://github.com/tidyverse/tidyverse/blob/main/R/logo.R

Examples

```
thisplot_logo()
```

to_percent *Convert proportions to percentages*

Description

Convert proportions to percentages

Usage

```
to_percent(x)
```

Arguments

x	A numeric vector.

Value

A numeric vector expressed as percentages.

Examples

```
to_percent(c(1, 1, 2))
to_percent(c(0, 0))
```

visual_colors *Visualize colors in HTML widget*

Description

Display a grid of color swatches with optional names or color codes.

Usage

```
visual_colors(
  colors,
  names = NULL,
  num_per_row = 30,
  title = NULL,
  label_mode = c("auto", "chinese", "pinyin", "rgb", "hex")
)
```

Arguments

colors	A character vector of hex color codes.
names	Optional. A character vector of names for each color. Default is NULL, which means hex color codes will be displayed. You can pass any labels (e.g., RGB values, custom names) via this parameter.
num_per_row	Number of colors per row. Default is 30.
title	Optional title for the visualization. Default is NULL.
label_mode	Label layout mode. Default is "auto", which detects the label type from names. Use "chinese", "pinyin", "rgb", or "hex" to force a layout optimized for that label style.

Value

An HTML widget.

Examples

```
# Visualize a simple color palette
visual_colors(
  colors = c("#FF0000", "#00FF00", "#0000FF"),
  names = c("Red", "Green", "Blue")
)

visual_colors(
  colors = c("#FF0000", "#00FF00"),
  names = c("(255, 0, 0)", "(0, 255, 0)")
)

visual_colors(thisplot::palette_list$Paired)

# Use with ChineseColors
cc <- ChineseColors()
visual_colors(
  colors = cc$blue[1:60],
  title = "Chinese Blue Colors"
)
```

Index

∗ **data**
chinese_colors, 15
palette_list, 54

add_grob, 4
adjcolors, 5
adjustlayout, 5
annotate_quadrants, 6
annotation_block_fill_graphics, 7
annotation_block_graphics, 8
annotation_graphics, 8
as_grob, 9
as_gtable, 9

Blend2Color, 10
blendcolors, 10
BlendRGBList, 11
build_heatmap_annotation, 12
build_patchwork, 13

chinese_colors, *14*, 15, *31*
ChineseColors, 13, *31*
cli::ansi_strip, *80*
cli::is_utf8_output, *80*
clip_symmetric_range, 16
cluster_within_group2, 16
col2hex, 17
ComplexHeatmap::anno_block, *7, 8*
ComplexHeatmap::anno_customize, *9*
ComplexHeatmap::HeatmapAnnotation, *12*
compute_velocity_on_grid, 17

DimDataPlot, 19
drop_data, 23

extractgrobs, 24

geom_alluvial, 24
geom_alluvial_label, 25
geom_alluvial_text
(geom_alluvial_label), 25
geom_sankey, 27
geom_sankey_bump, 28
geom_sankey_label, 29
geom_sankey_text (geom_sankey_label), 29
get_chinese_palettes, *14*, 30, *31*
get_colors, *14*, 31
get_legend, 32
get_vars, 33
ggplot2::geom_smooth, *62*
ggplot2::stat_ellipse, *21*
ggplot2::theme, *78*, *80*
GraphPlot, 33
grid::unit, *55*
grid::unit(), *58, 59*
grid_draw, 38

head(), *38*
head.colors, 38
heatmap_border_color
(heatmap_border_spec), 39
heatmap_border_enabled
(heatmap_border_spec), 39
heatmap_border_gp
(heatmap_border_spec), 39
heatmap_border_size_value
(heatmap_border_spec), 39
heatmap_border_spec, 39
heatmap_fixsize, 40
heatmap_legend_border
(heatmap_border_spec), 39
heatmap_rendersize, 41

invert_figures, 42
invert_png, 43
invert_svg, 43
invert_svg(), *66*

jitter_highlighted_points, 44

major_grid, 45

make_long, 46
mestimate, 46

NetworkPlot, 47
no_legend, 50
no_margin, 51
no_x_text, 51
no_y_text, 52
normalize_drawable, 50

palette_colors, 52, *69*
palette_list, *53*, 54, *69*
panel_fix, 54
panel_fix(), *59*
panel_fix_overall *(*panel_fix*)*, 54
patchwork_grob, 57
plot_autocompose, 57
plot_autosize, 59
plot_autosize(), *58*
plot_histogram, 60
plot_scatter, 61
print.ChineseColors, 63
print.ChineseColors(), *14*
print.colors, 63
print.colors(), *31*
print.thisplot_logo, 64
print_colored_table, 64

rasterise_plot, 65
RGBA2RGB, 66

segements_df, 67
show_palettes, *7*, *21*, *53*, *61*, *62*, 68, *70*
simple_colors, 69
slim_data, 70
standardise, 71
StatPlot, 71

theme_alluvial *(*theme_sankey*)*, 79
theme_blank, 77
theme_sankey, 79
theme_sankey_bump *(*theme_sankey*)*, 79
theme_this, 79
thisplot *(*thisplot-package*)*, 3
thisplot-package, 3
thisplot_logo, 80
to_percent, 81

visual_colors, *14*, 81