Package ‘cowplot’

July 11, 2019

Title Streamlined Plot Theme and Plot Annotations for 'ggplot2'

Version 1.0.0

Description Provides various features that help with creating publication-quality figures with 'ggplot2', such as a set of themes, functions to align plots and arrange them into complex compound figures, and functions that make it easy to annotate plots and mix plots with images. The package was originally written for internal use in the Wilke lab, hence the name (Claus O. Wilke's plot package). It has also been used extensively in the book Fundamentals of Data Visualization.

URL https://wilkelab.org/cowplot

BugReports https://github.com/wilkelab/cowplot/issues

Depends R (>= 3.5.0)

Imports ggplot2 (> 2.2.1), grid, gtable, grDevices, methods, rlang, scales, utils

License GPL-2

LazyData true

Suggests Cairo, covr, dplyr, forcats, gridGraphics (>= 0.4-0), knitr, lattice, magick, maps, PASWR, rmarkdown, testthat (>= 1.0.0), tidyr, vdiffr (>= 0.3.0), VennDiagram

VignetteBuilder knitr

Collate 'add_sub.R' 'align_plots.R' 'as_grob.R' 'as_gtable.R'
 'axis_canvas.R' 'cowplot.R' 'draw.R' 'get_plot_component.R'
 'get_axes.R' 'get_titles.R' 'get_legend.R' 'get_panel.R'
 'gtable.R' 'key_glyph.R' 'plot_grid.R' 'save.R'
 'set_null_device.R' 'setup.R' 'stamp.R' 'themes.R'
 'utils_ggplot2.R'

RoxygenNote 6.1.1

NeedsCompilation no

Author Claus O. Wilke [aut, cre] (<https://orcid.org/0000-0002-7470-9261>)

Maintainer Claus O. Wilke <wilke@austin.utexas.edu>

Repository CRAN

Date/Publication 2019-07-11 12:22:55 UTC
R topics documented:

add_sub .................................................. 3
align_plots ................................................. 4
as_grob .................................................... 6
as_gtable .................................................. 6
axis_canvas ............................................... 7
background_grid ........................................... 9
cowplot .................................................... 10
draw_figure_label ........................................ 10
draw_grob ................................................. 11
draw_image ................................................ 12
draw_label ............................................... 13
draw_line .................................................. 15
draw_plot .................................................. 16
draw_plot_label .......................................... 17
draw_text .................................................. 17
get_legend ................................................ 18
get_panel .................................................. 19
get_plot_component ...................................... 20
get_title ................................................... 21
get_y_axis ............................................... 21
ggdraw ..................................................... 22
ggsave2 .................................................... 23
gtable_remove_grobs .................................... 23
gtable_squash_cols ..................................... 24
gtable_squash_rows ..................................... 24
insert_xaxis_grob ....................................... 25
panel_border .............................................. 25
plot_grid .................................................. 26
png_null_device .......................................... 29
rectangle_key_glyph .................................... 30
save_plot ............................................... 31
set_null_device ......................................... 32
stamp ..................................................... 33
theme_cowplot .......................................... 34
theme_map ............................................... 35
theme_minimal_grid ..................................... 36
theme_nothing .......................................... 37

Index 39
Add annotation underneath a plot

Description

This function can add an arbitrary label or mathematical expression underneath the plot, similar to the `sub` parameter in base R. It is mostly superseded now by the `caption` argument to `ggplot2::labs()`, and it is recommended to use `caption` instead of `add_sub()` whenever possible.

Usage

```r
add_sub(plot, label, x = 0.5, y = 0.5, hjust = 0.5, vjust = 0.5,
         vpadding = grid::unit(1, "lines"), fontfamily = "",
         fontface = "plain", color = "black", size = 14, angle = 0,
         lineheight = 0.9, colour)
```

Arguments

- `plot` A `ggplot` object or `gtable` object derived from a `ggplot` object.
- `label` The label with which the plot should be annotated. Can be a plotmath expression.
- `x` The x position of the label
- `y` The y position of the label
- `hjust` Horizontal justification
- `vjust` Vertical justification
- `vpadding` Vertical padding. The total vertical space added to the label, given in grid units. By default, this is added equally above and below the label. However, by changing the `y` and `vjust` parameters, this can be changed.
- `fontfamily` The font family
- `fontface` The font face ("plain", "bold", etc.)
- `color`, `colour` Text color
- `size` Point size of text
- `angle` Angle at which text is drawn
- `lineheight` Line height of text

Details

The exact location where the label is placed is controlled by the parameters `x`, `y`, `hjust`, and `vjust`. By default, all these parameters are set to 0.5, which places the label centered underneath the plot panel. A value of `x = 0` indicates the left boundary of the plot panel and a value of `x = 1` indicates the right boundary. The parameter `hjust` works just as elsewhere in `ggplot2`. Thus, `x = 0`, `hjust = 0` places the label left-justified at the left boundary of the plot panel,
align_plots

Align multiple plots vertically and/or horizontally

Description

Align the plot area of multiple plots. Inputs are a list of plots plus alignment parameters. Horizontal or vertical alignment or both are possible. In the simplest case the function will align all elements of each plot, but it can handle more complex cases as long as the axis parameter is defined. In this
align_plots

In this case, alignment is done through a call to align_margin(). The function align_plots is called by the plot_grid() function and is usually not called directly, though direct calling of the function is useful if plots with multiple y-axes are desired (see example).

Usage

align_plots(..., plotlist = NULL, align = c("none", "h", "v", "hv"),
          axis = c("none", "l", "r", "t", "b", "lr", "tb", "tblr"),
          greedy = TRUE)

Arguments

... List of plots to be aligned.

plotlist (optional) List of plots to display. Alternatively, the plots can be provided individually as the first n arguments of the function align_plots (see plot_grid examples).

align (optional) Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. Options are align="none" (default), "hv" (align in both directions), "h", and "v".

axis (optional) Specifies whether graphs should be aligned by the left ("l"), right ("r"), top ("t"), or bottom ("b") margins. Options are axis="none" (default), or a string of any combination of "l", "r", "t", and/or "b" in any order (e.g. axis="tblr" or axis="rlbt" for aligning all margins)

greedy (optional) Defines the alignment policy when alignment axes are specified via the axis option. greedy = TRUE tries to always align by adjusting the outmost margin. greedy = FALSE aligns all columns/rows in the gtable if possible.

Examples

library(ggplot2)

p1 <- ggplot(mpg, aes(manufacturer, hwy)) + stat_summary(fun.y="median", geom = "bar") + theme_half_open() + theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust= 1))

p2 <- ggplot(mpg, aes(manufacturer, displ)) + geom_point(color="red") + scale_y_continuous(position = "right") + theme_half_open() + theme(axis.text.x = element_blank())

# manually align and plot on top of each other
aligned_plots <- align_plots(p1, p2, align="hv", axis="tblr")

# Note: In most cases two y-axes should not be used, but this example
# illustrates how one could accomplish it.
ggdraw(aligned_plots[[1]]) + draw_plot(aligned_plots[[2]])
as_grob

Convert a base plot or a ggplot2 plot into a grob

Description

This function does its best attempt to take whatever you provide it and turn it into a grob. It is primarily meant to convert ggplot plots into grobs, but it will also take any grid object (grob), a recorded base R plot, a formula specifying a base R plot, a function that generates a base R plot, or a trellis object.

Usage

as_grob(plot, device = NULL)

Arguments

plot The plot to convert
device A function that creates an appropriate null device. See set_null_device() for details. If set to NULL, will use the cowplot-wide default.

Examples

library(grid)
x <- 1:10
y <- (1:10)^2
p <- ~plot(x, y)
grid.newpage()
grid.draw(as_grob(p))

as_gtable

Convert plot or other graphics object into a gtable

Description

This function does its best attempt to take whatever you provide it and turn it into a gtable. It is primarily meant to convert ggplot plots into gtables, but it will also take any grid object (grob), a recorded R base plot, or a function that generates an R base plot.

Usage

as_gtable(plot)

plot_to_gtable(plot)
Axis canvas

Arguments

plot  The plot or other graphics object to convert into a gtable. Here, plot can be any object handled by `as_grob()`.

Details

To convert ggplot plots, the function needs to use a null graphics device. This can be set with `set_null_device()`.

Description

This function takes an existing ggplot2 plot and copies one or both of the axis into a new plot. The main idea is to use this in conjunction with `insert_xaxis_grob()` or `insert_yaxis_grob()` to draw custom axis-like objects or margin annotations. Importantly, while this function works for both continuous and discrete scales, notice that discrete scales are converted into continuous scales in the returned axis canvas. The levels of the discrete scale are placed at continuous values of 1, 2, 3, etc. See Examples for an example of how to convert a discrete scale into a continuous scale.

Usage

```
axis_canvas(plot, axis = "y", data = NULL, mapping = aes(),
           xlim = NULL, ylim = NULL, coord_flip = FALSE)
```

Arguments

plot  The plot defining the x and/or y axis range for the axis canvas.
axis  Specifies which axis to copy from plot. Can be "x", "y", or "xy".
data  (optional) Data to be displayed in this layer.
mapping  (optional) Aesthetic mapping to be used in this layer.
xlim  (optional) Vector of two numbers specifying the limits of the x axis. Ignored if the x axis is copied over from plot.
ylim  (optional) Vector of two numbers specifying the limits of the y axis. Ignored if the y axis is copied over from plot.
coord_flip  (optional) If true, flips the coordinate system and applies x limits to the y axis and vice versa. Useful in combination with ggplot2’s `coord_flip()` function.
Examples

# annotate line graphs with labels on the right
library(dplyr)
library(tidyrr)
library(ggplot2)
theme_set(theme_half_open())
x <- seq(0, 10, .1)
d <- data.frame(x,
    linear = x,
squared = x*x/5,
cubed = x*x*x/25)
gather(fun, y, -x)
pmain <- ggplot(d, aes(x, y, group = fun)) + geom_line() +
    scale_x_continuous(expand = c(0, 0))
paxis <- axis_canvas(pmain, axis = "y") +
    geom_text(data = filter(d, x == max(x)), aes(y = y, label = paste0(" ", fun)),
        x = 0, hjust = 0, vjust = 0.5)
ggdraw(insert_yaxis_grob(pmain, paxis, ggrid::unit(.25, "null")))

# discrete scale with integrated color legend
pmain <- ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
    geom_violin(trim = FALSE) + guides(fill = "none") +
    scale_x_discrete(labels = NULL) +
    theme_minimal()

label_data <- data.frame(x = 1:levels(iris$Species),
    Species = levels(iris$Species))
paxis <- axis_canvas(pmain, axis = "x", data = label_data, mapping = aes(x = x)) +
    geom_tile(aes(fill = Species, y = 0.5), width = 0.9, height = 0.3) +
    geom_text(aes(label = Species, y = 0.5), hjust = 0.5, vjust = 0.5, size = 11/pt)
ggdraw(insert_xaxis_grob(pmain, paxis, ggrid::unit(.07, "null"),
    position = "bottom")

# add marginal density distributions to plot
pmain <- ggplot(iris, aes(x=Sepal.Length, y=Sepal.Width, color=Species)) + geom_point()

xdens <- axis_canvas(pmain, axis = "x") +
    geom_density(data=iris, aes(x=Sepal.Length, fill=Species), alpha=0.7, size=.2)

# need to set `coord_flip` if you plan to use `coord_flip()`
ydens <- axis_canvas(pmain, axis = "y", coord_flip = TRUE) +
    geom_density(data=iris, aes(x=Sepal.Width, fill=Species), alpha=0.7, size=.2) +
    coord_flip()

p1 <- insert_xaxis_grob(pmain, xdens, ggrid::unit(.2, "null"), position = "top")
p2 <- insert_yaxis_grob(p1, ydens, ggrid::unit(.2, "null"), position = "right")
ggdraw(p2)
**background_grid**

Add/modify/remove the background grid in a ggplot2 plot

**Description**

This function provides a simple way to set the background grid in ggplot2. It doesn’t do anything that can’t be done just the same with `theme()`. However, it simplifies creation of the most commonly needed variations.

**Usage**

```r
background_grid(major = c("xy", "x", "y", "only_minor", "none"),
                minor = c("none", "xy", "x", "y"), size.major = 0.5,
                size.minor = 0.2, color.major = "grey85", color.minor = "grey85",
                colour.major, colour.minor)
```

**Arguments**

- `major`: Specifies along which axes you would like to plot major grid lines. Options are "xy", "x", "y", "none".
- `minor`: Specifies along which axes you would like to plot minor grid lines. Options are "xy", "x", "y", "none".
- `size.major`: Size of the major grid lines.
- `size.minor`: Size of the minor grid lines.
- `color.major`, `colour.major`: Color of the major grid lines.
- `color.minor`, `colour.minor`: Color of the minor grid lines.

**Details**

Note: This function completely overwrites all background grid settings of the current theme. If that is not what you want, you may be better off using `theme()` directly.

**Examples**

```r
library(ggplot2)

ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point() +
  theme_half_open() +
  background_grid()
```
**cowplot**  
*cowplot.*

**Description**

cowplot.

**draw_figure_label**  
*Add a label to a figure*

**Description**

The main purpose of this function is to add labels specifying extra information about the figure, such as "Figure 1", or "A" - often useful in cowplots with more than one pane. The function is similar to `draw_plot_label`.

**Usage**

```r
draw_figure_label(label, position = c("top.left", "top", "top.right", "bottom.left", "bottom", "bottom.right"), size, fontface, ...)
```

**Arguments**

- `label`  
  Label to be drawn

- `position`  
  Position of the label, can be one of "top.left", "top", "top.right", "bottom.left", "bottom", "bottom.right". Default is "top.left"

- `size`  
  (optional) Size of the label to be drawn. Default is the text size of the current theme

- `fontface`  
  (optional) Font face of the label to be drawn. Default is the font face of the current theme

- `...`  
  other arguments passed to `draw_plot_label`

**Author(s)**

Ulrik Stervbo (ulrik.stervbo @ gmail.com)

**See Also**

`draw_plot_label`
Examples

```r
df <- data.frame(
  x = 1:10, y1 = 1:10, y2 = (1:10)^2, y3 = (1:10)^3, y4 = (1:10)^4
)
p1 <- ggplot(df, aes(x, y1)) + geom_point()
p2 <- ggplot(df, aes(x, y2)) + geom_point()
p3 <- ggplot(df, aes(x, y3)) + geom_point()
p4 <- ggplot(df, aes(x, y4)) + geom_point()

# Create a simple grid
p <- plot_grid(p1, p2, p3, p4, align = 'hv')

# Default font size and position
p + draw_figure_label(label = "Figure 1")

# Different position and font size
p + draw_figure_label(label = "Figure 1", position = "bottom.right", size = 10)

# Using bold font face
p + draw_figure_label(label = "Figure 1", fontface = "bold")

# Making the label red and slanted
p + draw_figure_label(label = "Figure 1", angle = -45, colour = "red")

# Labeling an individual plot
ggdraw(p2) + draw_figure_label(label = "Figure 1", position = "bottom.right", size = 10)
```

draw_grob

**Draw a grob.**

Description

Places an arbitrary grob somewhere onto the drawing canvas. By default, coordinates run from 0 to 1, and the point (0, 0) is in the lower left corner of the canvas.

Usage

```r
draw_grob(grob, x = 0, y = 0, width = 1, height = 1, scale = 1, clip = "inherit", hjust = 0, vjust = 0)
```

Arguments

- **grob**: The grob to place.
- **x**: The x location of the grob. (Left side if hjust = 0.)
- **y**: The y location of the grob. (Bottom side if vjust = 0.)
width  Width of the grob.
height Height of the grob.
scale Scales the grob relative to the rectangle defined by x, y, width, height. A setting of scale = 1 indicates no scaling.
clip Set to "on" to clip the grob or "inherit" to not clip. Note that clipping doesn't always work as expected, due to limitations of the grid graphics system.
hjust Horizontal justification relative to x.
vjust Vertical justification relative to y.

Examples

# A grid grob (here a blue circle)
g <- grid::circleGrob(gp = grid::gpar(fill = "blue"))
# place into the middle of the plotting area, at a scale of 50%
+ ggdraw() + draw_grob(g, scale = 0.5)

Description

Places an image somewhere onto the drawing canvas. By default, coordinates run from 0 to 1, and the point (0, 0) is in the lower left corner of the canvas. Requires the magick package to work, and fails gracefully if that package is not installed.

Usage

draw_image(image, x = 0, y = 0, width = 1, height = 1, scale = 1,
clip = "inherit", interpolate = TRUE, hjust = 0, vjust = 0)

Arguments

image The image to place. Can be a file path, a URL, or a raw vector with image data, as in magick::image_read(). Can also be an image previously created by magick::image_read() and related functions.
x The x location of the image. (Left side if hjust = 0.)
y The y location of the image. (Bottom side if vjust = 0.)
width Width of the image.
height Height of the image.
scale Scales the image relative to the rectangle defined by x, y, width, height. A setting of scale = 1 indicates no scaling.
clip Set to "on" to clip the image relative to the box into which it is draw (useful for scale > 1). Note that clipping doesn't always work as expected, due to limitations of the grid graphics system.
interpolate  A logical value indicating whether to linearly interpolate the image (the alternative is to use nearest-neighbour interpolation, which gives a more blocky result).

ehjust  Horizontal justification relative to x.

evjust  Vertical justification relative to y.

Examples

```r
library(ggplot2)

# Use image as plot background
p <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
    geom_density(alpha = 0.7) +
    scale_y_continuous(expand = expand_scale(mult = c(0, 0.05))) +
    theme_half_open()

logo_file <- system.file("extdata", "logo.png", package = "cowplot")
ggdraw() +
    draw_image(logo_file, scale = .7) +
    draw_plot(p)

# Make grid with plot and image

cow_file <- system.file("extdata", "cow.jpg", package = "cowplot")
p2 <- ggdraw() + draw_image(cow_file, scale = 0.9)
plot_grid(p + theme(legend.position = c(1, 1), legend.justification = c(1, 1)),
          p2,
          labels = "AUTO")
```

# Manipulate images and draw in plot coordinates

```r
if (requireNamespace("magick", quietly = TRUE)){

    img <- magick::image_transparent(
        magick::image_read(logo_file),
        color = "white"
    )

    img2 <- magick::image_negate(img)

ggplot(data.frame(x = 1:3, y = 1:3), aes(x, y)) +
    geom_point(size = 3) +
    geom_abline(slope = 1, intercept = 0, linetype = 2, color = "blue") +
    draw_image(img , x = 1, y = 1, scale = .9) +
    draw_image(img2, x = 2, y = 2, scale = .9)
}
```

| draw_label | Draw a text label or mathematical expression. |
**Description**

This function can draw either a character string or mathematical expression at the given coordinates. It works both on top of ggdraw and directly with ggplot, depending on which coordinate system is desired (see examples).

**Usage**

```r
draw_label(label, x = 0.5, y = 0.5, hjust = 0.5, vjust = 0.5,
    fontfamily = "", fontface = "plain", color = "black", size = 14,
    angle = 0, lineheight = 0.9, alpha = 1, colour)
```

**Arguments**

- `label`: String or plotmath expression to be drawn.
- `x`: The x location (origin) of the label.
- `y`: The y location (origin) of the label.
- `hjust`: Horizontal justification. Default = 0.5 (centered on x). 0 = flush-left at x, 1 = flush-right.
- `vjust`: Vertical justification. Default = 0.5 (centered on y). 0 = baseline at y, 1 = ascender at y.
- `fontfamily`: The font family
- `fontface`: The font face ("plain", "bold", etc.)
- `color`, `colour`: Text color
- `size`: Point size of text
- `angle`: Angle at which text is drawn
- `lineheight`: Line height of text
- `alpha`: The alpha value of the text

**Details**

By default, the x and y coordinates specify the center of the text box. Set `hjust = 0`, `vjust = 0` to specify the lower left corner, and other values of `hjust` and `vjust` for any other relative location you want to specify.

**See Also**

- `ggdraw`

**Examples**

```r
library(ggplot2)

# setup plot and a label (regression description)
p <- ggplot(mtcars, aes(disp, mpg)) +
  geom_line(color = "blue") +
  theme_half_open()
```
draw_line

Draw a line from connected points

draw_line

Description

Provide a sequence of x values and accompanying y values to draw a line on a plot.

Usage

draw_line(x, y, ...)

Arguments

x Vector of x coordinates.

y Vector of y coordinates.

... geom_path parameters such as colour, alpha, size, etc.

Details

This is a convenience function, providing a wrapper around ggplot2’s geom_path.

See Also

gem_path, ggdraw

background_grid(minor = 'none')
out <- cor.test(mtcars$disp, mtcars$mpg, method = 'sp', exact = FALSE)
label <- substitute(
paste("Spearman ", rho, ", rho = ", estimate, ", P = ", pvalue),
list(estimate = signif(out$estimate, 2), pvalue = signif(out$p.value, 2)))

# Add label to plot, centered on {x,y} (in data coordinates)
p + draw_label(label, x = 300, y = 32)
# Add label to plot in data coordinates, flush-left at x, baseline at y.
p + draw_label(label, x = 100, y = 30, hjust = 0, vjust = 0)

# Add labels via ggdraw. Uses ggdraw coordinates.
# ggdraw coordinates default to xlim = c(0, 1), ylim = c(0, 1).
ggdraw(p) +
  draw_label("centered on 70\% of x range,\n90\% of y range", x = 0.7, y = 0.9)

ggdraw(p) +
  draw_label("bottom left at (0, 0)\", x = 0, y = 0, hjust = 0, vjust = 0) +
  draw_label("top right at (1, 1)\", x = 1, y = 1, hjust = 1, vjust = 1) +
  draw_label("centered on (0.5, 0.5)\", x = 0.5, y = 0.5, hjust = 0.5, vjust = 0.5)
Examples

```r
ggdraw() +
draw_line(
  x = c(0.2, 0.7, 0.7, 0.3),
  y = c(0.1, 0.3, 0.9, 0.8),
  color = "blue", size = 2
)
```

**draw_plot**  
*Draw a (sub)plot.*

Description

Places a plot somewhere onto the drawing canvas. By default, coordinates run from 0 to 1, and the point (0, 0) is in the lower left corner of the canvas.

Usage

```r
draw_plot(plot, x = 0, y = 0, width = 1, height = 1, scale = 1, hjust = 0, vjust = 0)
```

Arguments

- `plot` The plot to place. Can be a ggplot2 plot, an arbitrary grob or gtable, or a recorded base-R plot, as in `as_grob()`.
- `x` The x location of the plot. (Left side if `hjust = 0`.)
- `y` The y location of the plot. (Bottom side if `vjust = 0`.)
- `width` Width of the plot.
- `height` Height of the plot.
- `scale` Scales the grob relative to the rectangle defined by `x`, `y`, `width`, `height`. A setting of `scale = 1` indicates no scaling.
- `hjust` Horizontal justification relative to `x`.
- `vjust` Vertical justification relative to `y`.

Examples

```r
library(ggplot2)

# make a plot
p <- ggplot(data.frame(x = 1:3, y = 1:3), aes(x, y)) + geom_point()

# draw into the top-right corner of a larger plot area
ggdraw() + draw_plot(p, .6, .6, .4, .4)
```
**draw_plot_label**  
*Add a label to a plot*

**Description**

This function adds a plot label to the upper left corner of a graph (or an arbitrarily specified position). It takes all the same parameters as `draw_text`, but has defaults that make it convenient to label graphs with letters A, B, C, etc. Just like `draw_text()`, it can handle vectors of labels with associated coordinates.

**Usage**

```
draw_plot_label(label, x = 0, y = 1, hjust = -0.5, vjust = 1.5,
size = 16, fontface = "bold", family = NULL, color = NULL,
colour, ...)]
```

**Arguments**

- `label`  
  String (or vector of strings) to be drawn as the label.
- `x`  
  The x position (or vector thereof) of the label(s).
- `y`  
  The y position (or vector thereof) of the label(s).
- `hjust`  
  Horizontal adjustment.
- `vjust`  
  Vertical adjustment.
- `size`  
  Font size of the label to be drawn.
- `fontface`  
  Font face of the label to be drawn.
- `family`  
  (optional) Font family of the plot labels. If not provided, is taken from the current theme.
- `color`, `colour`  
  (optional) Color of the plot labels. If not provided, is taken from the current theme.
- `...`  
  Other arguments to be handed to `draw_text`.

**draw_text**  
*Draw multiple text-strings in one go.*

**Description**

This is a convenience function to plot multiple pieces of text at the same time. It cannot handle mathematical expressions, though. For those, use `draw_label`.

**Usage**

```
draw_text(text, x = 0.5, y = 0.5, size = 14, hjust = 0.5,
vjust = 0.5, ...)
```
get_legend

Arguments

- **text**: A vector of Character (not expressions) specifying the string(s) to be written.
- **x**: Vector of x coordinates.
- **y**: Vector of y coordinates.
- **size**: Font size of the text to be drawn.
- **hjust**: (default = 0.5)
- **vjust**: (default = 0.5)
- **...**: Style parameters, such as colour, alpha, angle, size, etc.

Details

Note that font sizes are scaled by a factor of 2.85, so sizes agree with those of the theme. This is different from `geom_text` in `ggplot2`.

By default, the x and y coordinates specify the center of the text box. Set `hjust = 0`, `vjust = 0` to specify the lower left corner, and other values of `hjust` and `vjust` for any other relative location you want to specify.

For a full list of `...` options, see `geom_label`.

See Also

draw_label

Examples

```r
# Draw onto a 1x1 drawing surface
ggdraw() + draw_text("Hello World!", x = 0.5, y = 0.5)
#
# Adorn a plot from the Anscombe data set of "identical" data.
library(ggplot2)

p <- ggplot(anscombe, aes(x1, y1)) + geom_point() + geom_smooth()
three_strings <- c("Hello World!", "to be or not to be", "over and out")
p + draw_text(three_strings, x = 8:10, y = 5:7, hjust = 0)
```

---

**get_legend**

*Retrieve the legend of a plot*

Description

This function extracts just the legend from a `ggplot`

Usage

get_legend(plot)
get_panel

Arguments

plot  A ggplot or gtable from which to retrieve the legend

Value

A gtable object holding just the legend or NULL if there is no legend.

Examples

library(ggplot2)
theme_set(theme_half_open())

p1 <- ggplot(mtcars, aes(mpg, disp)) + geom_line()
plot.mpg <- ggplot(mpg, aes(x = cty, y = hwy, colour = factor(cyl))) + geom_point(size=2.5)
# Note that these cannot be aligned vertically due to the legend in the plot.mpg
ggdraw(plot_grid(p1, plot.mpg, ncol=1, align='v'))

legend <- get_legend(plot.mpg)
plot.mpg <- plot.mpg + theme(legend.position='none')
# Now plots are aligned vertically with the legend to the right
ggdraw(plot_grid(plot_grid(p1, plot.mpg, ncol=1, align='v'),
                 plot_grid(NULL, legend, ncol=1),
                 rel_widths=c(1, 0.2)))

get_panel(Retrieve the panel or part of a panel of a plot)

Description

get_panel() extracts just the main panel from a ggplot or a specified panel in a faceted plot.
get_panel_component() extracts components from the panel, such as geoms.

Usage

get_panel(plot, panel = NULL, return_all = FALSE)
get_panel_component(panel, pattern)

Arguments

plot  A ggplot or gtable from which to retrieve the panel
panel  An integer indicating which panel to pull. ggplot orders panels column-wise, so this is in order from the top left down.
return_all  If there is more than one panel, should all be returned as a list? Default is FALSE.
pattern  the name of the component
get_plot_component

Value
A gtable object holding the panel(s) or a grob of the component

Examples
library(ggplot2)

p <- ggplot(mpg, aes(displ, cty)) + geom_point()
plot_panel <- get_panel(p)
ggdraw(plot_panel)

ggdraw(get_panel_component(plot_panel, "geom_point"))
get_title

Get plot titles

Description

These functions extract just the titles from a ggplot. get_title() pulls the title, while get_subtitle() pulls the subtitle.

Usage

get_title(plot)

get_subtitle(plot)

Arguments

plot

A ggplot or gtable.

Examples

library(ggplot2)

p <- ggplot(mpg, aes(displ, cty)) +
  geom_point() +
  labs(
    title = "Plot title",
    subtitle = "Plot subtitle"
  )
ggdraw(get_title(p))
ggdraw(get_subtitle(p))

get_y_axis

Get plot axes

Description

These functions extract just the axes from a ggplot. get_y_axis() pulls the y-axis, while get_x_axis() pulls the x-axis.

Usage

get_y_axis(plot, position = c("left", "right"))

get_x_axis(plot, position = c("bottom", "top"))
ggdraw

Set up a drawing layer on top of a ggplot.

Description

Set up a drawing layer on top of a ggplot.

Usage

ggdraw(plot = NULL, xlim = c(0, 1), ylim = c(0, 1), clip = "off")

Arguments

- **plot**: The plot to use as a starting point. Can be a ggplot2 plot, an arbitrary grob or gtable, or a recorded base-R plot, as in `as_grob()`.
- **xlim**: The x-axis limits for the drawing layer.
- **ylim**: The y-axis limits for the drawing layer.
- **clip**: Should drawing be clipped to the set limits? The default is no ("off").

Examples

```r
library(ggplot2)

p <- ggplot(mpg, aes(displ, cty)) + geom_point() + theme_minimal_grid()
ggdraw(p) + draw_label("Draft", colour = "#80404080", size = 120, angle = 45)
```
**Description**

This function behaves just like `ggsave()` from ggplot2. The main difference is that by default it doesn’t use the Dingbats font for pdf output. The Dingbats font causes problems with some pdf readers.

**Usage**

```r
ggsave2(filename, plot = ggplot2::last_plot(), device = NULL, path = NULL, scale = 1, width = NA, height = NA, units = c("in", "cm", "mm"), dpi = 300, limitsize = TRUE, ...)
```

**Arguments**

- `filename` (Filename of the plot.)
- `plot` (Plot to save, defaults to last plot displayed.)
- `device` (Device to use, automatically extract from file name extension.)
- `path` (Path to save plot to (if you just want to set path and not filename).)
- `scale` (Scaling factor.)
- `width` (Width (defaults to the width of current plotting window).)
- `height` (Height (defaults to the height of current plotting window).)
- `units` (Units for width and height when either one is explicitly specified (in, cm, or mm).)
- `dpi` (DPI to use for raster graphics.)
- `limitsize` (When TRUE (the default), ggsave2() will not save images larger than 50x50 inches, to prevent the common error of specifying dimensions in pixels.)
- `...` (Other arguments to be handed to the plot device.)

**gtable_remove_grobs**

*Remove named elements from gtable*

**Description**

Remove named elements from gtable

**Usage**

```r
gtable_remove_grobs(table, names, ...)
```
gtable_squash_rows

Arguments

table     The table from which grobs should be removed
names     A character vector of the grob names (as listed in \code{table$layout}) that should be removed
...       Other parameters passed through to \code{gtable_filter}.

---

gtable_squash_cols     \textit{Set the width of given columns to 0.}

Description

Set the width of given columns to 0.

Usage

\code{gtable_squash_cols(table, cols)}

Arguments

table     The gtable on which to operate
cols      Numerical vector indicating the columns whose width should be set to zero.

---

gtable_squash_rows     \textit{Set the height of given rows to 0.}

Description

Set the height of given rows to 0.

Usage

\code{gtable_squash_rows(table, rows)}

Arguments

table     The gtable on which to operate
rows      Numerical vector indicating the rows whose heights should be set to zero.
**insert_xaxis_grob**

*Insert an axis-like grob on either side of a plot panel in a ggplot2 plot.*

**Description**

The function `insert_xaxis_grob()` inserts a grob at the top or bottom of the plot panel in a `ggplot2` plot.

**Usage**

```r
insert_xaxis_grob(plot, grob, height = grid::unit(0.2, "null"),
                   position = c("top", "bottom"))
```

```r
insert_yaxis_grob(plot, grob, width = grid::unit(0.2, "null"),
                   position = c("right", "left"))
```

**Arguments**

- `plot`: The plot into which the grob will be inserted.
- `grob`: The grob to insert. This will generally have been obtained via `get_panel()` from a `ggplot2` object, in particular one generated with `axis_canvas()`. If a `ggplot2` plot is provided instead of a grob, then `get_panel()` is called to extract the panel grob.
- `height`: The height of the grob, in grid units. Used by `insert_xaxis_grob()`.
- `position`: The position of the grob. Can be "right" or "left" for `insert_yaxis_grob()` and "top" or "bottom" for `insert_xaxis_grob()`.
- `width`: The width of the grob, in grid units. Used by `insert_yaxis_grob()`.

**Details**

For usage examples, see `axis_canvas()`.

---

**panel_border**

*Add/remove the panel border in a ggplot2 plot*

**Description**

This function provides a simple way to modify the panel border in `ggplot2`. It doesn’t do anything that can’t be done just the same with `theme()`. However, it saves some typing.

**Usage**

```r
panel_border(color = "grey85", size = 1, linetype = 1,
             remove = FALSE, colour)
```
plot_grid

Arguments

- **color, colour** The color of the border.
- **size** Size. Needs to be twice as large as desired outcome when panel clipping is on (the default).
- **linetype** Line type.
- **remove** If TRUE, removes the current panel border.

**plot_grid** Arrange multiple plots into a grid

Description

Arrange multiple plots into a grid.

Usage

```r
plot_grid(..., plotlist = NULL, align = c("none", "h", "v", "hv"),
axis = c("none", "l", "r", "t", "b", "lr", "tb", "tblr"),
nrow = NULL, ncol = NULL, rel_widths = 1, rel_heights = 1,
labels = NULL, label_size = 14, label_fontfamily = NULL,
label_fontface = "bold", label_colour = NULL, label_x = 0,
label_y = 1, hjust = -0.5, vjust = 1.5, scale = 1,
greedy = TRUE, cols = NULL, rows = NULL)
```

Arguments

- ... List of plots to be arranged into the grid. The plots can be any objects that the function `as_gtable()` can handle (see also examples).
- **plotlist** (optional) List of plots to display. Alternatively, the plots can be provided individually as the first n arguments of the function `plot_grid` (see examples).
- **align** (optional) Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. Options are "none" (default), "hv" (align in both directions), "h", and "v".
- **axis** (optional) Specifies whether graphs should be aligned by the left ("l"), right ("r"), top ("t"), or bottom ("b") margins. Options are "none" (default), or a string of any combination of l, r, t, and b in any order (e.g. "tblr" or "rlbt" for aligning all margins). Must be specified if any of the graphs are complex (e.g. faceted) and alignment is specified and desired. See `align_plots()` for details.
- **nrow** (optional) Number of rows in the plot grid.
- **ncol** (optional) Number of columns in the plot grid.
- **rel_widths** (optional) Numerical vector of relative columns widths. For example, in a two-column grid, `rel_widths = c(2, 1)` would make the first column twice as wide as the second column.
`plot_grid`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rel_heights</code></td>
<td>(optional) Numerical vector of relative rows heights. Works just as <code>rel_widths</code> does, but for rows rather than columns.</td>
</tr>
<tr>
<td><code>labels</code></td>
<td>(optional) List of labels to be added to the plots. You can also set <code>labels=&quot;AUTO&quot;</code> to auto-generate upper-case labels or <code>labels=&quot;auto&quot;</code> to auto-generate lower-case labels.</td>
</tr>
<tr>
<td><code>label_size</code></td>
<td>(optional) Numerical value indicating the label size. Default is 14.</td>
</tr>
<tr>
<td><code>label_fontfamily</code></td>
<td>(optional) Font family of the plot labels. If not provided, is taken from the current theme.</td>
</tr>
<tr>
<td><code>label_fontface</code></td>
<td>(optional) Font face of the plot labels. Default is &quot;bold&quot;.</td>
</tr>
<tr>
<td><code>label_colour</code></td>
<td>(optional) Color of the plot labels. If not provided, is taken from the current theme.</td>
</tr>
<tr>
<td><code>label_x</code></td>
<td>(optional) Single value or vector of x positions for plot labels, relative to each subplot. Defaults to 0 for all labels. (Each label is placed all the way to the left of each plot.)</td>
</tr>
<tr>
<td><code>label_y</code></td>
<td>(optional) Single value or vector of y positions for plot labels, relative to each subplot. Defaults to 1 for all labels. (Each label is placed all the way to the top of each plot.)</td>
</tr>
<tr>
<td><code>hjust</code></td>
<td>Adjusts the horizontal position of each label. More negative values move the label further to the right on the plot canvas. Can be a single value (applied to all labels) or a vector of values (one for each label). Default is -0.5.</td>
</tr>
<tr>
<td><code>vjust</code></td>
<td>Adjusts the vertical position of each label. More positive values move the label further down on the plot canvas. Can be a single value (applied to all labels) or a vector of values (one for each label). Default is 1.5.</td>
</tr>
<tr>
<td><code>scale</code></td>
<td>Individual number or vector of numbers greater than 0. Enables you to scale the size of all or select plots. Usually it’s preferable to set margins instead of using <code>scale</code>, but <code>scale</code> can sometimes be more powerful.</td>
</tr>
<tr>
<td><code>greedy</code></td>
<td>(optional) How should margins be adjusted during alignment. See <code>align_plots()</code> for details.</td>
</tr>
<tr>
<td><code>cols</code></td>
<td>Deprecated. Use <code>ncol</code>.</td>
</tr>
<tr>
<td><code>rows</code></td>
<td>Deprecated. Use <code>nrow</code>.</td>
</tr>
</tbody>
</table>

Examples

```r
library(ggplot2)

df <- data.frame(
  x = 1:10, y1 = 1:10, y2 = (1:10)^2, y3 = (1:10)^3, y4 = (1:10)^4
)

p1 <- ggplot(df, aes(x, y1)) + geom_point()
p2 <- ggplot(df, aes(x, y2)) + geom_point()
p3 <- ggplot(df, aes(x, y3)) + geom_point()
p4 <- ggplot(df, aes(x, y4)) + geom_point()
p5 <- ggplot(mpg, aes(as.factor(year), hwy)) + geom_boxplot() +
```
```r
facet_wrap(~class, scales = "free_y")
# simple grid
plot_grid(p1, p2, p3, p4)

# simple grid with labels and aligned plots
plot_grid(
  p1, p2, p3, p4,
  labels = c('A', 'B', 'C', 'D'),
  align="hv"
)

# manually setting the number of rows, auto-generate upper-case labels
plot_grid(p1, p2, p3,
  nrow = 3,
  labels = "AUTO",
  label_size = 12,
  align = "v"
)

# making rows and columns of different widths/heights
plot_grid(
  p1, p2, p3, p4,
  align = 'hv',
  rel_heights = c(2,1),
  rel_widths = c(1,2)
)

# aligning complex plots in a grid
plot_grid(
  p1, p5,
  align = "h", axis = "b", nrow = 1, rel_widths = c(1,2)
)

# more examples

# missing plots in some grid locations, auto-generate lower-case labels
plot_grid(
  p1, NULL, NULL, p2, p3, NULL,
  ncol = 2,
  labels = "auto",
  label_size = 12,
  align = "v"
)

# can align top of plotting area as well as bottom
plot_grid(
  p1, p5,
  align = "h", axis = "tb",
  nrow = 1, rel_widths = c(1,2)
)

# other types of plots not generated with ggplot
p6 <- ~({
  plot_grid(p1, p2, p3, p4)
})
```
```r
par(
    mar = c(3, 3, 1, 1),
    mgp = c(2, 1, 0)
  )
plot(sqrt)
}

p7 <- function() {
  par(
    mar = c(2, 2, 1, 1),
    mgp = c(2, 1, 0)
  )
  image(volcano)
}
p8 <- grid::circleGrob()

plot_grid(p1, p6, p7, p8, labels = "AUTO", scale = c(1, .9, .9, .7))
```

---

**Description**

Null devices to be used when rendering graphics in the background. See `set_null_device()` for details.

**Usage**

```r
png_null_device(width, height)
```

```r
dpdf_null_device(width, height)
```

```r
cairo_null_device(width, height)
```

**Arguments**

- `width` Device width in inch
- `height` Device height in inch
Create customizable legend key glyphs

Description

These functions create customizable legend key glyphs, such as filled rectangles or circles.

Usage

rectangle_key_glyph(colour = NA, fill = fill, alpha = alpha,
size = size, linetype = linetype, padding = unit(c(0, 0, 0, 0), "pt"), color)

circle_key_glyph(colour = NA, fill = fill, alpha = alpha,
size = size, linetype = linetype, padding = unit(c(0, 0, 0, 0), "pt"), color)

Arguments

colour, color  Unquoted name of the aesthetic to use for the outline color, usually colour, color, or fill. Can also be a color constant, e.g. "red".

fill  Unquoted name of the aesthetic to use for the fill color, usually colour, color, or fill. Can also be a color constant, e.g. "red".

alpha  Unquoted name of the aesthetic to use for alpha, usually alpha. Can also be a numerical constant, e.g. 0.5.

size  Unquoted name of the aesthetic to use for the line thickness of the outline, usually size. Can also be a numerical constant, e.g. 0.5.

linetype  Unquoted name of the aesthetic to use for the line type of the outline, usually linetype. Can also be a constant, e.g. 2.

padding  Unit vector with four elements specifying the top, right, bottom, and left padding from the edges of the legend key to the edges of the key glyph.

Examples

library(ggplot2)

set.seed(1233)
df <- data.frame(
  x = sample(letters[1:2], 10, TRUE),
  y = rnorm(10)
)

ggplot(df, aes(x, y, color = x)) +
  geom_boxplot(
    key_glyph = rectangle_key_glyph(fill = color, padding = margin(3, 3, 3, 3))
  )
save_plot

ggplot(df, aes(x, y, color = x)) +
  geom_boxplot(
    key_glyph = circle_key_glyph(
      fill = color,
      color = "black", linetype = 3, size = 0.3,
      padding = margin(2, 2, 2)
    )
  )

save_plot

Alternative to ggsave(), with better support for multi-figure plots.

Description

This function replaces the standard ggsave() function for saving a plot into a file. It has several advantages over ggsave(). First, it uses default sizes that work well with the cowplot theme, so that frequently a plot size does not have to be explicitly specified. Second, it acknowledges that one often first develops individual plots and then combines them into multi-plot figures, and it makes it easy—in combination with plot_grid()—to carry out this workflow. Finally, it makes it easy to adjust the aspect ratio of the figure, which is frequently necessary to accommodate plots with or without figure legend.

Usage

save_plot(filename, plot, ncol = 1, nrow = 1, base_height = 3.71,
  base_asp = 1.618, base_width = NULL, ..., cols, rows,
  base_aspect_ratio, width, height)

Arguments

filename Name of the plot file to generate.
plot Plot to save.
ncol Number of subplot columns.
nrow Number of subplot rows.
base_height The height (in inches) of the plot or of one sub-plot if nrow or ncol > 1. Default is 3.71.
base_asp The aspect ratio (width/height) of the plot or of one sub-plot if nrow or ncol > 1. This argument is used if base_width = NULL or if base_height = NULL: if both width and height are provided then the aspect ratio is ignored. The default is 1.618 (the golden ratio), which works well for figures with a legend.
base_width The width (in inches) of the plot or of one sub-plot if nrow or ncol > 1. Default is NULL, which means that the width is calculated from base_height and base_aspect_ratio.
... Other arguments to be handed to ggsave2().
cols Deprecated. Use ncol.
The key idea for this function is that plots are often grids, with sub-plots at the individual grid locations. Therefore, for this function we specify a base width and aspect ratio that apply to one sub-plot, and we then specify how many rows and columns of subplots we have. This means that if we have code that can save a single figure, it is trivial to adapt this code to save a combination of multiple comparable figures. See examples for details.

Examples

```r
library(ggplot2)

# save a single plot with a legend
p1 <- ggplot(mpg, aes(x = cty, y = hwy, color = factor(cyl))) +
    geom_point(size = 2) +
    theme_half_open()
save_plot("p1.png", p1)

# same as p1 but determine base_width given base_height
save_plot("p2.png", p1, base_height = NULL, base_width = 6)

# save a single plot without legend, adjust aspect ratio
x <- (1:100)/10
p3 <- ggplot(data.frame(x = x, y = x*sin(x)), aes(x, y)) +
    geom_line() +
    theme_minimal_hgrid()
save_plot("p3.pdf", p3, base_asp = 1.1)

# now combine with a second plot and save
p3b <- ggplot(data.frame(x = x, y = cos(x)+x), aes(x, y)) +
    geom_line() +
    theme_minimal_hgrid()
p4 <- plot_grid(p3, p3b, labels = "AUTO")
save_plot("p4.pdf", p4, ncol = 2, base_asp = 1.1)
```

Description

The function `as_grob()` needs to open a graphics device to render ggplot objects into grid graphics objects. Unfortunately, there is no universally reliable graphics device available in R that always works. Therefore, this function allows you to switch out the null device.
Usage

set_null_device(null_device)

Arguments

null_device Either a string that defines the null device ("pdf", "png", "cairo") or a function that returns a new graphics device.

Details

You need to be aware that some graphics devices cause side effects when used as null devices. If you use an interactive device as null device, you may see an empty plot window pop up. Similarly, if you use a graphics device that writes a file, then you may find temporary files associated with the device. The default null device, pdf(NULL), does not cause these side effects. However, it has has other limitations. For example, on OS X, it cannot use all the fonts that are available on the system. The png device can use all fonts, but it will create temporary files.

See Also

Available null devices are: pdf_null_device(), png_null_device(), cairo_null_device()

Examples

set_null_device("png") # set the png null device

# create a jpeg null device
jpeg_null_device <- function(width, height) {
  jpeg(
    filename = tempfile(pattern = "jpeg_null_plot", fileext = ".jpg"),
    width = width, height = height, units = "in", res = 96
  )
  dev.control("enable")
}
set_null_device(jpeg_null_device)
theme_cowplot

Usage

```r
stamp(p, label, color = "black", alpha = 1, vjust = 1.1, hjust = 1,
size = 14, family = "", fontface = "bold", clip = "on", colour)

stamp_good(p, ...)

stamp_bad(p, ...)

stamp_wrong(p, ...)

stamp_ugly(p, ...)
```

Arguments

- `p` The plot to stamp
- `label` The text label used for the stamp
- `color, colour` The color of the stamp
- `alpha` Transparency level of the stamp
- `hjust, vjust` Horizontal and vertical adjustment of the label
- `size` Font size
- `family` Font family
- `fontface` Font face
- `clip` Should figure be clipped (default is "on")
- `...` Arguments handed off to `stamp()`.

Examples

```r
library(ggplot2)

p <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) +
geom_point(aes(color = factor(Petal.Width)))

stamp_bad(p + guides(color = "none"))

stamp_ugly(p)
```

theme_cowplot

Create the default cowplot theme

Description

The default cowplot theme, with a simple half-open frame and no grid. This theme used to be set by default after loading the cowplot package, but this is no longer the case.
theme_map

Usage

theme_cowplot(font_size = 14, font_family = "", line_size = 0.5,
   rel_small = 12/14, rel_tiny = 11/14, rel_large = 16/14)

theme_half_open(font_size = 14, font_family = "", line_size = 0.5,
   rel_small = 12/14, rel_tiny = 11/14, rel_large = 16/14)

Arguments

font_size Overall font size.
font_family Font family for plot title, axis titles and labels, legend texts, etc.
line_size Line size for axis lines.
rel_small Relative size of small text (e.g., axis tick labels)
rel_tiny Relative size of tiny text (e.g., caption)
rel_large Relative size of large text (e.g., title)

Details

Both theme_cowplot() and theme_half_open() provide exactly the same styling.

Value

The theme.

Examples

library(ggplot2)

ggplot(mtcars, aes(disp, mpg)) +
   geom_point() +
   theme_cowplot(font_size = 12)

theme_map

Create a theme for map plotting

Description

The theme created by this function is useful for plotting maps with cowplot default sizing.

Usage

theme_map(font_size = 14, font_family = "", line_size = 0.5,
   rel_small = 12/14, rel_tiny = 11/14, rel_large = 16/14)
Arguments

- `font_size` Overall font size. Default is 14.
- `font_family` Base font family.
- `line_size` Line size for axis lines.
- `rel_small` Relative size of small text (e.g., axis tick labels)
- `rel_tiny` Relative size of tiny text (e.g., caption)
- `rel_large` Relative size of large text (e.g., title)

Value

The theme.

Examples

```r
library(ggplot2)
library(maps)

usa_data = map_data("usa")
ggplot(usa_data, aes(long, lat, fill = region)) +
  geom_polygon() + theme_map()
ggplot(usa_data, aes(long, lat, fill = region)) +
  facet_wrap(~region, scales = "free") +
  geom_polygon() + theme_map()
```

theme_minimal_grid   Minimalistic themes with grids

Description

Three minimalistic themes that provide either a full grid, a horizontal grid, or a vertical grid. Similar to `theme_minimal()`, but with some stylistic differences. Most importantly, these themes do not draw minor grid lines. Also, font sizes are coordinated with `theme_half_open()` and with the defaults in the `save_plot()` function.

Usage

```r
theme_minimal_grid(font_size = 14, font_family = ",", line_size = 0.5,
  rel_small = 12/14, rel_tiny = 11/14, rel_large = 16/14,
  color = "grey85", colour)

theme_minimal_vgrid(font_size = 14, font_family = ",",
  line_size = 0.5, rel_small = 12/14, rel_tiny = 11/14,
  rel_large = 16/14, color = "grey85", colour)

theme_minimal_hgrid(font_size = 14, font_family = ",",
  line_size = 0.5, rel_small = 12/14, rel_tiny = 11/14,
  rel_large = 16/14, color = "grey85", colour)
```
Arguments

- **font_size**: Overall font size.
- **font_family**: Font family for plot title, axis titles and labels, legend texts, etc.
- **line_size**: Line size for grid lines.
- **rel_small**: Relative size of small text (e.g., axis tick labels)
- **rel_tiny**: Relative size of tiny text (e.g., caption)
- **rel_large**: Relative size of large text (e.g., title)
- **color, colour**: Color of grid lines.

Details

theme_minimal_grid() provides a minimal grid theme. theme_minimal_hgrid() strips down this theme even further and draws only horizontal lines, and theme_minimal_vgrid() does the same for vertical lines.

Examples

```r
library(ggplot2)

# theme_minimal_grid()
ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +
  geom_point() + theme_minimal_grid()

# theme_minimal_hgrid()
ggplot(mtcars, aes(x = carb)) +
  geom_bar(fill = "lightblue") +
  scale_y_continuous(limits = c(0, 11.5), expand = c(0, 0)) +
  theme_minimal_hgrid()

# theme_minimal_vgrid()
ggplot(mtcars, aes(x = carb)) +
  geom_bar(fill = "lightblue") +
  scale_y_continuous(limits = c(0, 11.5), expand = c(0, 0)) +
  coord_flip() +
  theme_minimal_vgrid()
```

---

**theme_nothing**

Create a completely empty theme

Description

The theme created by this function shows nothing but the plot panel.

Usage

```r
theme_nothing(font_size = 14, font_family = "", rel_small = 12/14)
```
Arguments

- font_size: Overall font size. Default is 14.
- font_family: Base font family.
- rel_small: Relative size of small text

Value

The theme.

Examples

```r
library(ggplot2)

ggplot(mtcars, aes(displ, mpg, color = cyl)) +
  geom_point() +
  theme_nothing()
```
Index

*Topic datasets
  draw_grob, 11

add_sub, 3
align_margin(), 5
align_plots(), 4
align_plots(), 26, 27
as_grob, 6
as_grob(), 7, 16, 22, 32
as_gtable, 6
as_gtable(), 26
axis_canvas, 7
axis_canvas(), 25

background_grid, 9

cairo_null_device (png_null_device), 29
cairo_null_device(), 33
circle_key_glyph (rectangle_key_glyph), 30
coord_flip(), 7
cowplot, 10
cowplot-package (cowplot), 10
draw_figure_label, 10
draw_grob, 11
draw_image, 12
draw_label, 13, 18
draw_line, 15
draw_plot, 16
draw_plot_label, 10, 17
draw_text, 17
gem_label, 18
gem_path, 15
GeomDrawGrob (draw_grob), 11
get_legend, 18
get_panel, 19
get_panel(), 25
get_panel_component (get_panel), 19
get_plot_component, 20
get_subtitle (get_title), 21
get_title, 21
g_x_axis (get_y_axis), 21
g_y_axis, 21
ggdraw, 14, 15, 22
ggplot2, 7, 25
ggplot2::labs(), 3
ggsave(), 23, 31
ggsave2, 23
ggsave2(). 31
gtble_remove_grobs, 23
table_squash_cols, 24
table_squash_rows, 24
insert_xaxis_grob, 25
insert_xaxis_grob(), 7
insert_yaxis_grob (insert_xaxis_grob), 25
insert_yaxis_grob(), 7
panel_border, 25
pdf_null_device (png_null_device), 29
pdf_null_device(), 33
plot_component_names
  (get_plot_component), 20
plot_components (get_plot_component), 20
plot_grid, 26
plot_grid(), 5, 31
plot_to_gtable (as_gtable), 6
png_null_device, 29
png_null_device(), 33
rectangle_key_glyph, 30

save_plot, 31
save_plot(), 36
set_null_device, 32
set_null_device(), 6, 7, 29
stamp, 33
stamp_bad (stamp), 33

39
stamp_good(stamp), 33
stamp_ugly(stamp), 33
stamp_wrong(stamp), 33

tHEME(): 9
theme_cowplot, 34
theme_half_open(theme_cowplot), 34
theme_half_open(), 36
theme_map, 35
theme_minimal(), 36
theme_minimal_grid, 36
theme_minimal_hgrid
(theme_minimal_grid), 36
theme_minimal_vgrid
(theme_minimal_grid), 36
theme_nothing, 37