Package ‘cowplot’

July 15, 2018

Title Streamlined Plot Theme and Plot Annotations for ‘ggplot2’

Version 0.9.3

Description Some helpful extensions and modifications to the ‘ggplot2’ package. In particular, this package makes it easy to combine multiple ‘ggplot2’ plots into one and label them with letters, e.g. A, B, C, etc., as is often required for scientific publications. The package also provides a streamlined and clean theme that is used in the Wilke lab, hence the package name, which stands for Claus O. Wilke’s plot package.

URL https://github.com/wilkelab/cowplot

Depends R (>= 3.3.0), ggplot2 (>= 2.1.0),

Imports grid (>= 3.0.0), gtable (>= 0.1.2), plyr (>= 1.8.2), grDevices, methods, scales, utils

License GPL-2

LazyData true

Suggests covr, gridGraphics, knitr, magick, maps, dplyr, tidyr, testthat, vdiffr, viridis

VignetteBuilder knitr

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


RoxygenNote 6.0.1.9000

NeedsCompilation no

Author Claus O. Wilke [aut, cre],
RStudio [cph] (Copyright for ggplot2 code copied to cowplot)

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

Repository CRAN

Date/Publication 2018-07-15 14:30:03 UTC
**R topics documented:**

- `add_sub` .......................................................... 2
- `align_margin` .................................................... 4
- `align_plots` ..................................................... 5
- `axis_canvas` .................................................... 6
- `background_grid` ................................................ 7
- `cowplot` ......................................................... 8
- `draw_figure_label` ................................................ 8
- `draw_grob` ....................................................... 10
- `draw_image` ...................................................... 10
- `draw_label` ...................................................... 12
- `draw_line` ....................................................... 13
- `draw_plot` ....................................................... 14
- `draw_plot_label` ................................................ 15
- `draw_text` ....................................................... 15
- `get_legend` ...................................................... 16
- `get_panel` ....................................................... 17
- `ggdraw` .......................................................... 18
- `ggsave` .......................................................... 18
- `gtable_remove_grobs` ........................................... 19
- `gtable_squash_cols` .......................................... 20
- `gtable_squash_rows` .......................................... 20
- `insert_xaxis_grob` ............................................. 21
- `panel_border` ................................................... 21
- `plot_grid` ........................................................ 22
- `plot_to_gtable` .................................................. 24
- `save_plot` ....................................................... 25
- `theme_cowplot` .................................................. 26
- `theme_map` ...................................................... 27
- `theme_nothing` .................................................. 27

**Index** 29

---

**add_sub**

*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",
        colour = "black", size = 14, angle = 0, lineheight = 0.9)
```
**add_sub**

**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.)
- **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, x = 0.5, hjust = 0.5 places the label centered underneath the plot panel, and x = 1, hjust = 1 places it right-justified at the right boundary of the plot panel. x-values below 0 or above 1 are allowed, and they move the label beyond the limits of the plot panel.

The y coordinates are relative to the added vertical space that is introduced underneath the x-axis label to place the annotation. A value of y=0 indicates the bottom-most edge of that space and a value of y=1 indicates the top-most edge of that space. The total height of the added space is given by the height needed to draw the label plus the value of vpadding. Thus, if y=0, vjust=0 then the extra padding is added entirely above the label, if y=1, vjust=1 then the extra padding is added entirely below the label, and if y=0.5, vjust=0.5 (the default) then the extra padding is added equally above and below the label. As is the case with x, y-values outside the range 0-1 are allowed. In particular, for sufficiently large values of y, the label will eventually be located inside the plot panel.

**Value**

A gtable object holding the modified plot.
Examples

```r
p1 <- ggplot(mtcars, aes(mpg, disp)) + geom_line(colour = "blue") + background_grid(minor = "none")
ggdraw(add_sub(p1, "This is an annotation.\nAnnotations can span multiple lines."))

# You can also do this repeatedly.
p2 <- add_sub(p1, "This formula has no relevance here: ", y = 0, vjust = 0)
p3 <- add_sub(p2, expression(paste(a^2+b^2, " = ", c^2)))
ggdraw(p3)

# This code also works with faceted plots:
plot.iris <- ggplot(iris, aes(Sepal.Length, Sepal.Width)) +
  geom_point() + facet_grid(. ~ Species) + stat_smooth(method = "lm") +
  background_grid(major = "y", minor = "none") + # add thin horizontal lines
  panel.border() # and a border around each panel
p2 <- add_sub(plot.iris, "Annotation underneath a faceted plot, left justified.", x = 0, hjust = 0)
ggdraw(p2)

# Finally, it is possible to move the annotation inside of the plot if desired.
ggdraw(add_sub(p1, "Annotation inside plot", vpadding = grid::unit(0, "lines"),
  y = 6, x = 0.03, hjust = 0))
```

align_margin

Align multiple plots along a specified margin

Description

The function aligns the dimensions of multiple plots along a specified axis, and is solely a helper function for `align_plots()` to reduce redundancy. Each element of the `sizes` list corresponds to the dimensions of a plot being aligned. They should be vectors created from calls to `grob$heights` or `grob$widths` depending on whether you are aligning vertically or horizontally. The list of dimensions is generated automatically by the `align_plots()` function, but see examples. If the same number of elements exist for all plots for the specified margin, the function will align individual elements on the margin. Otherwise, it aligns the plot by adding white space to plot margins so that all margins have the same dimensions.

Usage

```r
align_margin(sizes, margin_to_align)
```

Arguments

- `sizes`: list of dimensions for each plot being aligned. Each element of list obtained by a call to `grob$heights` or `grob$widths` (see example).
- `margin_to_align`: string either "first" or "last" for which part of plot area should be aligned. If vertically aligning, "first" aligns left margin and "last" aligns right margin. If horizontally aligning "first" aligns top margin and "last" aligns bottom margin.
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 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"))

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)
**axis_canvas**

Generates a canvas onto which one can draw axis-like objects.

### 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

```r
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

```r
p1 <- ggplot(mpg, aes(manufacturer, hwy)) + 
  stat_summary(fun.y="median", geom = "bar") + 
  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(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 would could accomplish it.
ggdraw(aligned_plots[[1]]) + draw_plot(aligned_plots[[2]])
```
Examples

# annotate line graphs with labels on the right
library(dplyr)
library(tidyr)
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 = paste(" ", fun)), x = 0, hjust = 0, vjust = 0.5)
ggdraw(insert_yaxis_grob(pmain, paxis, grid::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:nlevels(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, grid::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)
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, grid::unit(.2, "null"), position = "top")
p2 <- insert_yaxis_grob(p1, ydens, grid::unit(.2, "null"), position = "right")
ggdraw(p2)
**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, ...)  
```
draw_figure_label

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
p1 <- qplot(1:10, 1:10)
p2 <- qplot(1:10, (1:10)^2)
p3 <- qplot(1:10, (1:10)^3)
p4 <- qplot(1:10, (1:10)^4)

# 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**

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

**Arguments**

- **grob**: The grob to place.
- **x**: The x location of the lower left corner of the grob.
- **y**: The y location of the lower left corner of the grob.
- **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.

**Examples**

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

**draw_image**

*Draw an image*

**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.
draw_image

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

Arguments

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

Arguments

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

Arguments

image The image to place. Can be a file path, a URL, or a raw vector with image

x The x location of the lower left corner of the image.

y The y location of the lower left corner of the image.

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

clip Set to "on" to clip the image relative to the box into which it is draw (useful

interpolate A logical value indicating whether to linearly interpolate the image (the alterna-

Examples

# Use image as plot background
p <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) + geom_density(alpha = 0.7)
ggdraw() +
  draw_image("http://jeroen.github.io/images/tiger.svg") +
  draw_plot(p + theme(legend.box.background = element_rect(color = "white")))

# Make grid with plot and image
p <- ggplot(iris, aes(x = Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.7)
p2 <- ggdraw() + draw_image("http://jeroen.github.io/images/tiger.svg", scale = 0.9)
plot_grid(p, p2, labels = "AUTO")

# Manipulate images and draw in plot coordinates
if (requireNamespace("magick", quietly = TRUE)){
  img <- magick::image_read("http://jeroen.github.io/images/tiger.svg")
  img <- magick::image_transparent(img, 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

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

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.)
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

# setup plot and a label (regression description)
p <- ggplot(mtcars, aes(mpg, disp)) + geom_line(color = "blue") + background_grid(minor = 'none')
c <- cor.test(mtcars$mpg, mtcars$disp, method = 'sp')
label <- substitute(paste("Spearman ", rho, ", estimate ", P = ", pvalue),
                 list(estimate = signif(c$estimate, 2), pvalue = signif(c$p.value, 2)))

# Add label to plot, centered on {x,y} (in data coordinates)
p + draw_label(label, x = 20, y = 400)
# Add label to plot in data coordinates, flush-left at x, baseline at y.
p + draw_label(label, x = 20, y = 400, hjust = 0, vjust = 0)

# Add label to plot. Data coordinates, drawing rightward
# from x, with ascenders of text touching y.
p + draw_label(label, x = 20, y = 400, hjust = 0, vjust = 1)

# 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, 90% of y height", x = 0.7, y = 0.9)
labstr = "bottom left at (0%, 0%) of the SHEET, not the plot!"
p = ggdraw(p) + draw_label(labstr, x = 0, y = 0, hjust = 0, vjust = 0)
p = p + draw_label("top right at \{1,1\}", x = 1, y = 1, hjust = 1, vjust = 1)
p = p + draw_label("bottom left at \{.4,.4\}", x = 0.4, y = 0.4, hjust = 0, vjust = 0)
p + draw_label("centered on at \{.5,.5\}", x = 0.5, y = 0.5, hjust = 0.5, vjust = 0.5)

draw_line

Draw a line from connected points

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

geom_path, ggdraw
draw_plot

**Draw a (sub)plot.**

draw_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**

draw_plot(plot, x = 0, y = 0, width = 1, height = 1, scale = 1)

**Arguments**

- **plot**: The plot to place. Can be a ggplot2 plot, an arbitrary grob or gtable, or a recorded base-R plot, as in `plot_to_gtable()`.
- **x**: The x location of the lower left corner of the plot.
- **y**: The y location of the lower left corner of the plot.
- **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.

**Examples**

```r
# make a plot
p <- qplot(1:10, 1:10)
# 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, colour = NULL, ...)

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.
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

Description

This function extracts just the legend from a ggplot.

Usage

get_legend(plot)

Arguments

plot A ggplot or gtable from which to retrieve the legend

draw_label

Examples

# Draw onto a 1*1 drawing surface
ggdraw() + draw_text("Hello World!", x = 0.5, y = 0.5)

# Adorn a plot from the Anscombe data set of "identical" data.
p = qplot(x = x1, y = y1, geom = c("smooth", "point"), data = anscombe)
threeStrings = c("Hello World!", "to be or not to be", "over and out")
p + draw_text(threeStrings, x = 8:10, y = 5:7, hjust = 0)

get_legend

Retrieve the legend of a plot

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

Description

This function extracts just the legend from a ggplot.

Usage

get_legend(plot)

Arguments

plot A ggplot or gtable from which to retrieve the legend
get_panel

Value

A gtable object holding just the legend

Examples

```r
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)))
```

Description

This function extracts just the main panel from a ggplot. It only works for plots with exactly one panel (i.e., plots that are not faceted).

Usage

```r
get_panel(plot)
```

Arguments

- `plot`: A ggplot or gtable from which to retrieve the panel

Value

A gtable object holding just the panel
**ggdraw**

*Set up a drawing layer on top of a ggplot*

**Description**

Set up a drawing layer on top of a ggplot.

**Usage**

```r
ggdraw(plot = NULL, xlim = c(0, 1), ylim = c(0, 1))
```

**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 `plot_to_gtable()`.  
- `xlim`: The x-axis limits for the drawing layer.  
- `ylim`: The y-axis limits for the drawing layer.

**Examples**

```r
p <- ggplot(mpg, aes(displ, cty)) + geom_point()
ggdraw(p) + draw_label("Draft", colour = "grey", size = 120, angle = 45)
```

---

**ggsave**

*Cowplot reimplementation of ggsave.*

**Description**

This function should behave just like ggsave from ggplot2, with the main difference being that by default it doesn’t use the Dingbats font for pdf output. If you ever have trouble with this function, you can use `ggplot2::ggsave()` instead.

**Usage**

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

**Arguments**

- `filename`  
  Filename of 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), ggsave 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**

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

**Arguments**

- `table`  
  The table from which grobs should be removed

- `names`  
  A character vector of the grob names (as listed in `table$layout`) that should be removed

- `...`  
  Other parameters passed through to `gtable_filter`. 
gtable_squash_cols

Set the width of given columns to 0.

Description
Set the width of given columns to 0.

Usage

gtable_squash_cols(table, cols)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The gtable on which to operate</td>
</tr>
<tr>
<td>cols</td>
<td>Numerical vector indicating the columns whose width should be set to zero.</td>
</tr>
</tbody>
</table>

---

gtable_squash_rows

Set the height of given rows to 0.

Description
Set the height of given rows to 0.

Usage

gtable_squash_rows(table, rows)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>table</td>
<td>The gtable on which to operate</td>
</tr>
<tr>
<td>rows</td>
<td>Numerical vector indicating the rows whose heights should be set to zero.</td>
</tr>
</tbody>
</table>
**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. The function `insert_yaxis_grob()` inserts a grob to the right or left of the plot panel in a ggplot2 plot.

**Usage**

```r
code
```

```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
code
```

```r
panel_border(colour = "gray80", size = 0.5, linetype = 1,
             remove = FALSE)
```
plot_grid

Arguments

- **colour**: The color of the border.
- **size**: Size.
- **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, cols = NULL, rows = NULL)
```

**Arguments**

- ...: List of plots to be arranged into the grid. The plots can be any objects that the function `plot_to_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.
- **rel_heights**: (optional) Numerical vector of relative columns heights. Works just as rel_widths does, but for rows rather than columns.
plot_grid

labels (optional) List of labels to be added to the plots. You can also set labels="AUTO" to auto-generate upper-case labels or labels="auto" to auto-generate lower-case labels.

label_size (optional) Numerical value indicating the label size. Default is 14.

label_fontfamily (optional) Font family of the plot labels. If not provided, is taken from the current theme.

label_fontface (optional) Font face of the plot labels. Default is "bold".

label_colour (optional) Color of the plot labels. If not provided, is taken from the current theme.

label_x (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.)

label_y (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.)

hjust 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.

vjust 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.

scale 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 scale, but scale can sometimes be more powerful.

cols Deprecated. Use ncol.

rows Deprecated. Use nrow.

Examples

```r
p1 <- qplot(1:10, 1:10)
p2 <- qplot(1:10, (1:10)^2)
p3 <- qplot(1:10, (1:10)^3)
p4 <- qplot(1:10, (1:10)^4)
p5 <- ggplot(mpg, aes(as.factor(year), hwy)) +
  geom_boxplot() +
  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_to_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

plot_to_gtable(plot)

Arguments

plot The plot or other graphics object to convert into a gtable. Here, plot can be an object of the following classes: ggplot, recordedplot, grob, or gtable. Alternatively, plot can be a function creating a plot when called (see examples for plot_grid()).
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 the figure legend.

Usage

```r
save_plot(filename, plot, ncol = 1, nrow = 1, base_height = 4,
          base_aspect_ratio = 1.1, base_width = NULL, ..., cols = NULL,
          rows = NULL)
```

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 4.
- **base_aspect_ratio** The aspect ratio 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 width or height is missing the aspect ratio will be used calculate the NULL value. The default is 1.1, which works well for figures without 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 height and `base_aspect_ratio`.
- **...** Other arguments to be handed to `ggsave`.
- **cols** Deprecated. Like `ncol`.
- **rows** Deprecated. Like `nrow`.

Details

The key idea for this function is that plots are often grids, with sup-plots at the individual grid locations. Therefore, for this function we specify a base width and aspect ratio that apply to one sup-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

# save a single plot without legend
x <- (1:100)/10
p1 <- qplot(x, 2*x+5, geom='line')
save_plot("p1.pdf", p1)
# now combine with a second plot and save
p2B <- qplot(x, -x^2+10*x-3, geom='line')
p2 <- plot_grid(p1, p2B, labels=c("A", "B"))
save_plot("p2.pdf", p2, ncol = 2)
# save a single plot with legend, changing the aspect ratio to make room for the legend
p3 <- ggplot(mpg, aes(x = cty, y = hwy, colour = factor(cyl))) + geom_point(size=2.5)
save_plot("p3.png", p3, base_aspect_ratio = 1.3)
# same as p3 but determine base_height given base_aspect_ratio and base_width
p4 <- ggplot(mpg, aes(x = cty, y = hwy, colour = factor(cyl))) + geom_point(size=2.5)
save_plot("p4.png", p4, base_height = NULL, base_aspect_ratio = 1.618, base_width = 6)
# same as p4 but determine base_width given base_aspect_ratio and base_height
p5 <- ggplot(mpg, aes(x = cty, y = hwy, colour = factor(cyl))) + geom_point(size=2.5)
save_plot("p5.png", p5, base_height = 6, base_aspect_ratio = 1.618, base_width = NULL)

---

theme_cowplot

Create the default cowplot theme

Description

After loading the cowplot package, this theme will be the default for all graphs made with ggplot2.

Usage

theme_cowplot(font_size = 14, font_family = "", line_size = 0.5)

Arguments

- font_size: Overall font size. Default is 14.
- font_family: Default font family.
- line_size: Default line size.

Value

The theme.

Examples

qplot(1:10, (1:10)^2) + theme_cowplot(font_size = 15)
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(base_size = 14, base_family = "")

Arguments
base_size   Overall font size. Default is 14.
base_family Base font family.

Value
The theme.

Examples
usa_data = map_data("usa")
ggplot(usa_data, aes(long, lat, group=region)) + geom_polygon() + theme_map()
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_nothing

Create a completely empty theme

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

Usage
theme_nothing(base_size = 14, base_family = "")

Arguments
base_size   Overall font size. Default is 14.
base_family Base font family.

Value
The theme.
Examples

qplot(1:10, (1:10)^2) + theme_nothing()
Index

*Topic datasets
  draw_grob, 10
add_sub, 2
align_margin, 4
align_margin(), 5
align_plots, 5
align_plots(), 4, 22
axis_canvas, 6
axis_canvas(), 21
background_grid, 7
coord_flip(), 6
cowplot, 8
cowplot-package (cowplot), 8
draw_figure_label, 8
draw_grob, 10
draw_image, 10
draw_label, 12, 16
draw_line, 13
draw_plot, 14
draw_plot_label, 9, 15
draw_text, 15
geom_label, 16
geom_path, 13
GeomDrawGrob (draw_grob), 10
get_legend, 16
get_panel, 17
get_panel(), 21
ggdraw, 12, 13, 18
ggplot, 24
ggplot2, 6, 21
ggplot2::labs(), 2
ggsave, 18
grob, 24
gtable, 24
gtable_remove_grobs, 19
gtable_squash_cols, 20
gtable_squash_rows, 20
insert_xaxis_grob, 21
insert_xaxis_grob(), 6
insert_xaxis_grob(insert_xaxis_grob), 21
panel_border, 21
plot_grid, 22
plot_grid(), 5, 24
plot_to_gtable, 24
plot_to_gtable(), 14, 18, 22
recordedplot, 24
save_plot, 25
theme_cowplot, 26
theme_map, 27
theme_nothing, 27

29