

Package ‘cowplot’

July 30, 2017

Title Streamlined Plot Theme and Plot Annotations for 'ggplot2'

Version 0.8.0

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, utils

License GPL-2

LazyData true

Suggests knitr, gridGraphics, viridis

VignetteBuilder knitr

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

Collate 'add_sub.R' 'cowplot.R' 'draw.R' 'get_legend.R' 'gtable.R'
'plot_grid.R' 'save.R' 'setup.R' 'switch_axis.R' 'themes.R'

RoxygenNote 6.0.1

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 2017-07-30 05:21:28 UTC

R topics documented:

add_sub	2
align_margin	4
align_plots	5
background_grid	6
cowplot	6
draw_figure_label	7
draw_grob	8
draw_label	8
draw_line	10
draw_plot	10
draw_plot_label	11
draw_text	11
get_legend	12
ggdraw	13
ggsave	13
gtable_remove_grobs	14
gtable_squash_cols	15
gtable_squash_rows	15
panel_border	16
plot_grid	16
save_plot	18
theme_cowplot	20
theme_nothing	20
Index	22

add_sub	<i>Add annotation underneath a plot</i>
---------	---

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arbitrary label or mathematical expression underneath the plot, similar to the sub parameter in base R. It is mostly superseded now by the caption argument to `ggplot2::labs()`, and it is recommended to use `caption` instead of `add_sub()` whenever possible.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

```
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)
```

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

```
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

```
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 <code>grob\$heights</code> or <code>grob\$widths</code> (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

Examples

```
# Example for how to utilize, though align_plots() does this internally and automatically
p1 <- qplot(1:10, 1:10)
p2 <- qplot(1:10, (1:10)^2)
p3 <- qplot(1:10, (1:10)^3)
plots <- list(p1, p2, p3)
grobs <- lapply(plots, ggplot2::ggplotGrob)
plot_widths <- lapply(grobs, function(x){x$widths})
# Aligning the Left margins of all plots
aligned_widths <- align_margin(plot_widths, "first")
# Aligning the right margins of all plots as well
aligned_widths <- align_margin(plot_widths, "last")
# Setting the dimensions of plots to the aligned dimensions
for(i in 1:3){
  plots[[i]]$widths <- aligned_widths[[i]]
}
```

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

<code>...</code>	List of plots to be aligned.
<code>plotlist</code>	(optional) List of plots to display. Alternatively, the plots can be provided individually as the first n arguments of the function <code>align_plots</code> (see <code>plot_grid</code> examples).
<code>align</code>	(optional) Specifies whether graphs in the grid should be horizontally ("h") or vertically ("v") aligned. Options are <code>align="none"</code> (default), "hv" (align in both directions), "h", and "v".
<code>axis</code>	(optional) Specifies whether graphs should be aligned by the left ("l"), right ("r"), top ("t"), or bottom ("b") margins. Options are <code>axis="none"</code> (default), or a string of any combination of "l", "r", "t", and/or "b" in any order (e.g. <code>axis="tblr"</code> or <code>axis="rlbt"</code> for aligning all margins)

Examples

```
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]])
```

background_grid	<i>Add/modify/remove the background grid in a ggplot2 plot</i>
-----------------	--

Description

This function provides a simple way to modify 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

```
background_grid(major = c("xy", "x", "y", "only_minor", "none"),
  minor = c("xy", "x", "y", "none"), size.major = 0.2, size.minor = 0.5,
  colour.major = "grey90", colour.minor = "grey98")
```

Arguments

major	Specifies along which axes you would like to plot major grid lines. Options are "xy", "x", "y", "only_minor" (disables major grid lines but allows you to switch on minor grid lines), "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.
colour.major	Color of the major grid lines.
colour.minor	Color of the minor grid lines.

cowplot	<i>cowplot.</i>
---------	-----------------

Description

cowplot.

draw_figure_label	<i>Add a label to a figure</i>
-------------------	--------------------------------

Description

This function is similar to `draw_plot_label`, just with slightly different arguments and defaults. The main purpose of this function is to add labels specifying extra information about the figure, such as "Figure 1", which is sometimes useful.

Usage

```
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 <code>draw_plot_label</code>

Author(s)

Ulrik Stervbo (ulrik.stervbo @ gmail.com)

Examples

```
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.

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 of the label.
y	The y location of the label.
hjust	Horizontal justification
vjust	Vertical justification
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.

Examples

```
p <- ggplot(mtcars, aes(mpg, disp)) + geom_line(colour = "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)))
# adding label via ggdraw, in the ggdraw coordinates
ggdraw(p) + draw_label(label, .7, .9)
# adding label directly to plot, in the data coordinates
p + draw_label(label, 20, 400, hjust = 0, vjust = 0)
```

draw_line	<i>Draw a line.</i>
-----------	---------------------

Description

This is a convenience function. It's just a thin wrapper around `geom_line`.

Usage

```
draw_line(x, y, ...)
```

Arguments

x	Vector of x coordinates.
y	Vector of y coordinates.
...	Style parameters, such as colour, alpha, size, etc.

draw_plot	<i>Draw a (sub)plot.</i>
-----------	--------------------------

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 gtable, or a recorded base-R plot, as in plot_grid() .
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 <code>scale = 1</code> indicates no scaling.

draw_plot_label	<i>Add a label to a plot</i>
-----------------	------------------------------

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 <code>draw_text</code> .

draw_text	<i>Draw text.</i>
-----------	-------------------

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, ...)
```

Arguments

text	Character or expression vector specifying the text to be written.
x	Vector of x coordinates.
y	Vector of y coordinates.
size	Font size of the text to be drawn.
...	Style parameters, such as colour, alpha, angle, size, etc.

Details

Note that font sizes get scaled by a factor of 2.85, so sizes given here agree with font sizes used in 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.

get_legend	<i>Retrieve the legend of a plot</i>
------------	--------------------------------------

Description

This function extracts just the legend from a `ggplot`

Usage

```
get_legend(plot)
```

Arguments

plot	A <code>ggplot</code> or <code>gtable</code> from which to retrieve the legend
------	--

Value

A `gtable` object holding just the legend

Examples

```
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)))
```

ggdraw	<i>Set up a drawing layer on top of a ggplot</i>
--------	--

Description

Set up a drawing layer on top of a ggplot

Usage

```
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 gtable, or a recorded base-R plot, as in <code>plot_grid()</code> .
xlim	The x-axis limits for the drawing layer.
ylim	The y-axis limits for the drawing layer.

Examples

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

ggsave	<i>Cowplot reimplementaion of ggsave.</i>
--------	---

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 try using `ggplot2::ggsave()` instead.

Usage

```
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, ...)
```

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 <code>table\$layout</code>) that should be removed
...	Other parameters passed through to <code>gtable_filter</code> .

`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

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

`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

<code>table</code>	The gtable on which to operate
<code>rows</code>	Numerical vector indicating the rows whose heights should be set to zero.

panel_border	<i>Add/remove the panel border in a ggplot2 plot</i>
--------------	--

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

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

Arguments

colour	The color of the border.
size	Size.
linetype	Line type.
remove	If TRUE, removes the current panel border.

plot_grid	<i>Arrange multiple plots into a grid</i>
-----------	---

Description

Arrange multiple plots into a grid.

Usage

```
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 objects of one of the following classes: ggplot , recordedplot , gtable , or alternative can be a function creating a plot when called (see 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, <code>rel_widths = c(2, 1)</code> would make the first column twice as wide as the second column.
rel_heights	(optional) Numerical vector of relative columns heights. Works just as <code>rel_widths</code> does, but for rows rather than columns.
labels	(optional) List of labels to be added to the plots. You can also set <code>labels="AUTO"</code> to auto-generate upper-case labels or <code>labels="auto"</code> 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) 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) 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. Default is -0.5.
vjust	Adjusts the vertical position of each label. More positive values move the label further down on the plot canvas. 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 <code>ncol</code> .
rows	Deprecated. Use <code>nrow</code> .

Examples

```
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")

# 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")

# 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))

# 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
dev.new()
par(xpd = NA, # switch off clipping, necessary to always see axis labels
  bg = "transparent", # switch off background to avoid obscuring adjacent plots
  oma = c(2, 2, 0, 0), # move plot to the right and up
  mgp = c(2, 1, 0) # move axis labels closer to axis
)
plot(sqrt)
p6 <- recordPlot()
dev.off()
p7 <- function() image(volcano)
p8 <- gtable::gtable_col("circle", list(grid::circleGrob()))

plot_grid(p1, p6, p7, p8, labels = "AUTO", scale = c(1, 1, .85, .9))

```

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

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

Arguments

<code>filename</code>	Name of the plot file to generate.
<code>plot</code>	Plot to save.
<code>ncol</code>	Number of subplot columns.
<code>nrow</code>	Number of subplot rows.
<code>base_height</code>	The height (in inches) of the plot or of one sub-plot if <code>nrow</code> or <code>ncol</code> > 1. Default is 4.
<code>base_aspect_ratio</code>	The aspect ratio of the plot or of one sub-plot if <code>nrow</code> or <code>ncol</code> > 1. This argument is used if <code>base_width</code> = <code>NULL</code> or if <code>base_height</code> = <code>NULL</code> ; if width or height is missing the aspect ratio will be used calculate the <code>NULL</code> value. The default is 1.1, which works well for figures without a legend.
<code>base_width</code>	The width (in inches) of the plot or of one sub-plot if <code>nrow</code> or <code>ncol</code> > 1. Default is <code>NULL</code> , which means that the width is calculated from height and <code>base_aspect_ratio</code> .
<code>...</code>	Other arguments to be handed to <code>ggsave</code> .
<code>cols</code>	Deprecated. Like <code>ncol</code> .
<code>rows</code>	Deprecated. Like <code>nrow</code> .

Details

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

```
# 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	<i>Create the default cowplot theme</i>
---------------	---

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_nothing	<i>Create a completely empty theme</i>
---------------	--

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, 8
- add_sub, 2
- align_margin, 4
- align_margin(), 5
- align_plots, 5
- align_plots(), 4, 17
- background_grid, 6
- cowplot, 6
- cowplot-package (cowplot), 6
- draw_figure_label, 7
- draw_grob, 8
- draw_label, 8
- draw_line, 10
- draw_plot, 10
- draw_plot_label, 11
- draw_text, 11
- GeomDrawGrob (draw_grob), 8
- get_legend, 12
- ggdraw, 13
- ggplot, 16
- ggplot2::labs(), 2
- ggsave, 13
- gtable, 16
- gtable_remove_grobs, 14
- gtable_squash_cols, 15
- gtable_squash_rows, 15
- panel_border, 16
- plot_grid, 16
- plot_grid(), 5, 10, 13
- recordedplot, 16
- save_plot, 18
- theme_cowplot, 20
- theme_nothing, 20