

Package ‘see’

August 1, 2019

Type Package

Title Visualisation Toolbox for 'easystats' and Extra Geoms, Themes and Color Palettes for 'ggplot2'

Version 0.2.1

Date 2019-08-01

Maintainer Daniel Lüdecke <d.luedecke@uke.de>

URL <https://easystats.github.io/see/>

BugReports <https://github.com/easystats/see/issues>

Description Provides plotting utilities supporting easystats-packages (<<https://github.com/easystats/easystats>>) and some extra themes, geoms, and scales for 'ggplot2'. Color scales are based on <<https://www.materialui.co/colors>>.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.0), graphics, grDevices, stats

Imports bayestestR, dplyr, ggplot2, ggridges, grid, gridExtra, insight, magrittr, rlang, scales

Suggests emmeans, ggraph, ggrepel, MASS, rmarkdown, rstanarm, tidygraph, tidyr

RoxygenNote 6.1.1

Language en-GB

NeedsCompilation no

Author Daniel Lüdecke [aut, cre] (<<https://orcid.org/0000-0002-8895-3206>>),
Dominique Makowski [aut, inv] (<<https://orcid.org/0000-0001-5375-9967>>),
Philip Waggoner [aut, ctb] (<<https://orcid.org/0000-0002-7825-7573>>),
Mattan S. Ben-Shachar [aut] (<<https://orcid.org/0000-0002-4287-4801>>)

Repository CRAN

Date/Publication 2019-08-01 20:00:02 UTC

R topics documented:

add_plot_attributes	2
data_plot	3
flat_colors	7
geom_point2	8
geom_poolpoint	9
geom_violindot	10
geom_violinhalf	11
how_to_plot	12
material_colors	13
metro_colors	14
palette_flat	14
palette_material	15
palette_metro	15
palette_pizza	16
palette_social	16
pizza_colors	17
plots	17
scale_color_flat	18
scale_color_material	19
scale_color_metro	21
scale_color_pizza	22
scale_color_social	23
social_colors	25
theme_abbyss	25
theme_blackboard	26
theme_lucid	28
theme_modern	29
Index	31

add_plot_attributes *Complete figure with its attributes*

Description

The [data_plot](#) function usually stores information (such as title, axes labels etc.) as attributes. This function adds those information to the plot.

Usage

```
add_plot_attributes(x)
```

Arguments

x An object.

Examples

```
## Not run:
library(rstanarm)
library(bayestestR)
library(see)
library(ggplot2)

model <- stan_glm(
  Sepal.Length ~ Petal.Width + Species + Sepal.Width,
  data = iris,
  chains = 2, iter = 200
)

result <- hdi(model, ci = c(0.5, 0.75, 0.9, 0.95))
data <- data_plot(result, data = model)

p <- data %>%
  ggplot(aes(x = x, y = y, height = height, group = y, fill = fill)) +
  ggridges::geom_ridgeline_gradient()

p
p + add_plot_attributes(data)
## End(Not run)
```

data_plot

Prepare objects for plotting or plot objects

Description

data_plot() attempts to extract and transform an object to be further plotted, while plot() tries to visualize results of functions from different packages of the [easystats-project](#).

Usage

```
data_plot(x, data = NULL, ...)

## S3 method for class 'see_bayesfactor_models'
plot(x, n_pies = c("one", "many"),
     value = c("none", "BF", "probability"), log = FALSE,
     prior_odds = NULL, ...)

## S3 method for class 'see_bayesfactor_parameters'
plot(x, point_size = 2,
     rope_color = "#0171D3", rope_alpha = 0.2, show_intercept = FALSE,
     ...)
```

```

## S3 method for class 'see_check_distribution'
plot(x, point_size = 2, panel = TRUE,
     ...)

## S3 method for class 'see_check_normality'
plot(x, type = c("density", "qq", "pp"),
     data = NULL, ...)

## S3 method for class 'see_check_outliers'
plot(x, text_size = 3.5, ...)

## S3 method for class 'see_equivalence_test'
plot(x, rope_color = "#0171D3",
     rope_alpha = 0.2, show_intercept = FALSE, ...)

## S3 method for class 'see_estimate_density'
plot(x, stack = TRUE,
     show_intercept = FALSE, ...)

## S3 method for class 'see_parameters_model'
plot(x, show_intercept = FALSE,
     dot_size = 0.8, ...)

## S3 method for class 'see_parameters_sem'
plot(x, data = NULL,
     type = c("regression", "correlation", "loading"),
     threshold_coefficient = NULL, threshold_p = NULL, ci = TRUE, ...)

## S3 method for class 'see_point_estimate'
plot(x, data = NULL, point_size = 2,
     text_size = 3.5, panel = TRUE, show_labels = TRUE,
     show_intercept = FALSE, ...)

## S3 method for class 'see_rope'
plot(x, data = NULL, rope_alpha = 0.5,
     rope_color = "cadetblue", show_intercept = FALSE, ...)

```

Arguments

x	An object.
data	The original data used to create this object. Can be a statistical model or such.
...	Arguments passed to or from other methods.
n_pies	Number of pies.
value	What value to display.
log	Show log-transformed Bayes factors.
prior_odds	optional vector of prior odds for the models. See <code>BayesFactor::priorOdds</code> . As the size of the pizza slices corresponds to posterior probability (which is a

	function of prior probability and the BF), custom prior_odds will change the slices' size.
point_size	Size of point-geoms.
rope_color	Color of ROPE ribbon.
rope_alpha	Transparency level of ROPE ribbon.
show_intercept	Logical, if TRUE, the intercept-parameter is included in the plot. By default, it is hidden because in many cases the intercept-parameter has a posterior distribution on a very different location, so density curves of posterior distributions for other parameters are hardly visible.
panel	Logical, if TRUE, plots are arranged as panels; else, single plots are returned.
type	Character vector, indicating the type of plot (for <code>check_normality</code> or <code>parameters::model_parameters</code>
text_size	Size of text labels.
stack	Logical, if TRUE, densities are plotted as stacked lines. Else, densities are plotted for each parameter among each other.
dot_size	Size of point geoms.
threshold_coefficient	Numeric, threshold at which value coefficients will be displayed.
threshold_p	Numeric, threshold at which value p-values will be displayed.
ci	Logical, whether confidence intervals should be added to the plot.
show_labels	Logical, if TRUE, the text labels for the point estimates (i.e. <i>"Mean"</i> , <i>"Median"</i> and/or <i>"MAP"</i>) are shown. You may set <code>show_labels = FALSE</code> in case of overlapping labels, and add your own legend or footnote to the plot.

Details

`data_plot()` is in most situation not needed when the purpose is plotting, since most `plot()`-functions in **see** internally call `data_plot()` to prepare the data for plotting.

Many `plot()`-functions have a `data`-argument that is needed when the data or model for plotting can't be retrieved via `data_plot()`. In such cases, `plot()` gives an error and asks for providing data or models.

Most `plot()`-functions work out-of-the-box, i.e. you don't need to do much more than calling `plot(<object>)` (see 'Examples'). Some `plot`-functions allow to specify arguments to modify the transparency or color of geoms, these are shown in the 'Usage' section.

Plot-functions are available for objects from following functions (note that functions from packages might be listed here that are currently still in development and probably not yet available):

- `bayestestR::bayesfactor_models()`
- `bayestestR::bayesfactor_parameters()`
- `bayestestR::bayesfactor_savagedickey()`
- `bayestestR::ci()`
- `bayestestR::equivalence_test()`

- `bayestestR::estimate_density()`
- `bayestestR::eti()`
- `bayestestR::hdi()`
- `bayestestR::map_estimate()`
- `bayestestR::p_direction()`
- `bayestestR::point_estimate()`
- `bayestestR::rope()`
- `estimate::estimate_contrasts()`
- `parameters::model_parameters()`
- `performance::binned_residuals()`
- `performance::check_collinearity()`
- `performance::check_distribution()`
- `performance::check_heteroscedasticity()`
- `performance::check_homogeneity()`
- `performance::check_model()`
- `performance::check_normality()`
- `performance::check_outliers()`
- `performance::performance_roc()`

See Also

[Package-Vignettes](#)

Examples

```
## Not run:
library(bayestestR)
library(rstanarm)

model <- stan_glm(
  Sepal.Length ~ Petal.Width * Species,
  data = iris,
  chains = 2, iter = 200
)

x <- rope(model)
plot(x)

x <- hdi(model)
plot(x) + theme_modern()

data <- rnorm(1000, 1)
x <- p_direction(data)
plot(x)
```

```
x <- p_direction(model)
plot(x)

model <- stan_glm(
  mpg ~ wt + gear + cyl + disp,
  chains = 2,
  iter = 200,
  data = mtcars
)
x <- equivalence_test(model)
plot(x)
## End(Not run)

library(bayestestR)
library(see)

lm0 <- lm(qsec ~ 1, data = mtcars)
lm1 <- lm(qsec ~ drat, data = mtcars)
lm2 <- lm(qsec ~ wt, data = mtcars)
lm3 <- lm(qsec ~ drat + wt, data = mtcars)

result <- bayesfactor_models(lm1, lm2, lm3, denominator = lm0)

plot(result, n_pies = "one", value = "probability") + theme_modern() +
  scale_fill_pizza(reverse = TRUE)
```

flat_colors

Extract Flat UI colors as hex codes

Description

Can be used to get the hex code of specific colors from the Flat UI color palette. Use `flat_colors()` to see all available color.

Usage

```
flat_colors(...)
```

Arguments

... Character names of colors.

Value

A character vector with color-codes.

Examples

```
flat_colors()

flat_colors("dark red", "teal")
```

geom_point2	<i>Better looking points</i>
-------------	------------------------------

Description

Somewhat nicer points (especially in case of transparency) without borders and contour.

Usage

```
geom_point2(..., stroke = 0, shape = 16)

geom_jitter2(..., size = 2, stroke = 0, shape = 16)
```

Arguments

...	Other arguments to be passed to geom_point.
stroke	Stroke thickness.
shape	Shape of points.
size	Size of points.

Examples

```
library(ggplot2)
library(see)

normal <- ggplot(iris, aes(x = Petal.Width, y = Sepal.Length)) +
  geom_point(size = 8, alpha = 0.3) +
  theme_modern()

new <- ggplot(iris, aes(x = Petal.Width, y = Sepal.Length)) +
  geom_point2(size = 8, alpha = 0.3) +
  theme_modern()

plots(normal, new, ncol=2)
```

geom_poolpoint	<i>Pool ball points</i>
----------------	-------------------------

Description

Points labelled with the observation name.

Usage

```
geom_poolpoint(label, size_text = 3.88, size_background = size_text *  
2, size_point = size_text * 3.5, ...)
```

```
geom_pooljitter(label, size_text = 3.88, size_background = size_text *  
2, size_point = size_text * 3.5, jitter = 0.1, ...)
```

Arguments

label	Label to add inside the points.
size_text	Size of text.
size_background	Size of the white background circle.
size_point	Size of the ball.
...	Other arguments to be passed to geom_point.
jitter	Width and height of position jitter.

Examples

```
library(ggplot2)  
library(see)  
  
ggplot(iris, aes(x = Petal.Width, y = Sepal.Length, color = Species )) +  
  geom_poolpoint(label = rownames(iris)) +  
  scale_color_flat_d() +  
  theme_modern()  
  
ggplot(iris, aes(x = Petal.Width, y = Sepal.Length, color = Species )) +  
  geom_pooljitter(label = rownames(iris)) +  
  scale_color_flat_d() +  
  theme_modern()
```

geom_violindot	<i>Half-violin Half-dot plot</i>
----------------	----------------------------------

Description

Create a half-violin half-dot plot, useful for visualising the distribution and the sample size at the same time.

Usage

```
geom_violindot(mapping = NULL, data = NULL, trim = TRUE,
  scale = "area", show.legend = NA, inherit.aes = TRUE,
  size_dots = 0.7, color_dots = NULL, fill_dots = NULL,
  binwidth = 0.05, position_dots = ggplot2::position_nudge(x = -0.025,
  y = 0), ...)
```

Arguments

mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).
trim	If <code>TRUE</code> (default), trim the tails of the violins to the range of the data. If <code>FALSE</code> , don't trim the tails.
scale	if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.
show.legend	logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
size_dots	Size adjustment for dots.
color_dots	Color adjustment for dots.
fill_dots	Fill adjustment for dots.

binwidth	When method is "dotdensity", this specifies maximum bin width. When method is "histodot", this specifies bin width. Defaults to 1/30 of the range of the data
position_dots	Position adjustment for dots, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired geom/stat.

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_violindot() +
  theme_modern()
```

geom_violinhalf	<i>Half-violin plot</i>
-----------------	-------------------------

Description

Create a half-violin plot.

Usage

```
geom_violinhalf(mapping = NULL, data = NULL, stat = "ydensity",
  position = "dodge", trim = TRUE, scale = "area",
  show.legend = NA, inherit.aes = TRUE, ...)
```

Arguments

mapping	Set of aesthetic mappings created by <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).

<code>stat</code>	The statistical transformation to use on the data for this layer, as a string.
<code>position</code>	Position adjustment, either as a string, or the result of a call to a position adjustment function.
<code>trim</code>	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don't trim the tails.
<code>scale</code>	if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.
<code>show.legend</code>	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
<code>inherit.aes</code>	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
<code>...</code>	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired <code>geom/stat</code> .

See Also

<https://stackoverflow.com/questions/52034747/plot-only-one-side-half-of-the-violin-plot>

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_violinhalf() +
  theme_modern() +
  scale_fill_material_d()
```

how_to_plot

How to plot your object

Description

Access the source code used for plotting.

Usage

```
how_to_plot(x, ...)
```

Arguments

x An object.
... Arguments passed to or from other methods.

Details

The plotting-capability of the **see** package mainly evolve around two functions: `data_plot()`, which prepares the data from various objects to bring it into shape for plotting, and `plot()`, which takes the data from `data_plot()` and creates the ggplot-object. Although ggplot-objects are easily modifiable, it is sometimes necessary to build up a plot from scratch, using the data that should be plotted. This is where `how_to_plot()` can help. It simply extracts and polishes the code from the various `plot()` methods and prints it to the console. This code can be used as "basis" for building own ggplots.

Examples

```
## Not run:  
library(bayestestR)  
results <- hdi(rnorm(1000))  
  
how_to_plot(results)  
  
## End(Not run)
```

material_colors	<i>Extract material design colors as hex codes</i>
-----------------	--

Description

Can be used to get the hex code of specific colors from the material design color palette. Use `material_colors()` to see all available color.

Usage

```
material_colors(...)
```

Arguments

... Character names of colors.

Value

A character vector with color-codes.

Examples

```
material_colors()
material_colors("indigo", "lime")
```

metro_colors	<i>Extract Metro colors as hex codes</i>
--------------	--

Description

Can be used to get the hex code of specific colors from the Metro color palette. Use `metro_colors()` to see all available color.

Usage

```
metro_colors(...)
```

Arguments

... Character names of colors.

Value

A character vector with color-codes.

Examples

```
metro_colors()
metro_colors("dark red", "teal")
```

palette_flat	<i>Flat UI color palette</i>
--------------	------------------------------

Description

The palette based on Flat UI colors (<https://www.materialui.co/flatuicolors>).

Usage

```
palette_flat(palette = "contrast", reverse = FALSE, ...)
```

Arguments

palette	Character name of palette. Can be "full", "ice", "rainbow", "complement", or "contrast".
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Details

This function is usually not called directly, but from within `scale_color_flat()`.

palette_material	<i>Material design color palette</i>
------------------	--------------------------------------

Description

The palette based on material design colors (<https://www.materialui.co/colors>).

Usage

```
palette_material(palette = "contrast", reverse = FALSE, ...)
```

Arguments

palette	Character name of palette. Can be "full", "ice", "rainbow", "complement", or "contrast".
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Details

This function is usually not called directly, but from within `scale_color_material()`.

palette_metro	<i>Metro color palette</i>
---------------	----------------------------

Description

The palette based on Metro colors (<https://www.materialui.co/metrocolors>).

Usage

```
palette_metro(palette = "complement", reverse = FALSE, ...)
```

Arguments

palette	Character name of palette. Can be "full", "ice", "rainbow", "complement", or "contrast".
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Details

This function is usually not called directly, but from within `scale_color_metro()`.

palette_pizza	<i>Pizza color palette</i>
---------------	----------------------------

Description

The palette based on authentic neapolitan pizzas.

Usage

```
palette_pizza(palette = "margherita", reverse = FALSE, ...)
```

Arguments

palette	Pizza type. Can be "margherita" (default), "margherita_crust", "diavola" or "diavola_crust".
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Details

This function is usually not called directly, but from within `scale_color_pizza()`.

palette_social	<i>Social color palette</i>
----------------	-----------------------------

Description

The palette based on Social colors (<https://www.materialui.co/socialcolors>).

Usage

```
palette_social(palette = "complement", reverse = FALSE, ...)
```


Arguments

palette	Character name of palette. Can be "full", "ice", "rainbow", "complement", or "contrast".
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Details

This function is usually not called directly, but from within `scale_color_social()`.

pizza_colors	<i>Extract pizza colors as hex codes</i>
--------------	--

Description

Extract pizza colors as hex codes

Usage

```
pizza_colors(...)
```

Arguments

...	Character names of pizza ingredients.
-----	---------------------------------------

Value

A character vector with color-codes.

plots	<i>Multiple plots side by side</i>
-------	------------------------------------

Description

A wrapper around `gridExtra::grid.arrange` to plot multiple figures side by side on the same page.

Usage

```
plots(..., nrow = NULL, ncol = NULL, tags = FALSE)
```

Arguments

...	grobs, gtables, ggplot or trellis objects
nrow	argument of gtable
ncol	argument of gtable
tags	Add tags to your subfigures. Can be FALSE (no tags), TRUE (letter tags) or character vector containing tags labels.

Examples

```
library(ggplot2)
library(see)

p1 <- ggplot(iris, aes(x=Petal.Length, y=Sepal.Width)) + geom_point()
p2 <- ggplot(iris, aes(x=Petal.Length)) + geom_density()

plots(p1, p2)
plots(p1, p2, ncol=2, tags=TRUE)
plots(p1, p2, ncol=2, tags=c("Fig. 1", "Fig. 2"))
```

scale_color_flat *Flat UI color palette*

Description

The palette based on Flat UI (<https://www.materialui.co/flatuicolors>). Use `scale_color_flat_d` for *discrete* categories and `scale_color_flat_c` for a *continuous* scale.

Usage

```
scale_color_flat(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_color_flat_d(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_color_flat_c(palette = "contrast", discrete = FALSE,
  reverse = FALSE, ...)

scale_colour_flat(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_colour_flat_c(palette = "contrast", discrete = FALSE,
  reverse = FALSE, ...)

scale_colour_flat_d(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_fill_flat(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_fill_flat_d(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_fill_flat_c(palette = "contrast", discrete = FALSE,
  reverse = FALSE, ...)
```

Arguments

palette	Character name of palette. Can be "full", "ice", "rainbow", "complement", or "contrast".
discrete	Boolean indicating whether color aesthetic is discrete or not.
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments passed to <code>discrete_scale()</code> or <code>scale_color_gradientn()</code> , used respectively when <code>discrete</code> is TRUE or FALSE.

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  theme_modern() +
  scale_fill_flat_d()

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_violin() +
  theme_modern() +
  scale_fill_flat_d(palette = "ice")

ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Sepal.Length)) +
  geom_point() +
  theme_modern() +
  scale_color_flat_c(palette = "rainbow")
```

scale_color_material *Material design color palette*

Description

The palette based on material design colors (<https://www.materialui.co/color>). Use `scale_color_material_d()` for *discrete* categories and `scale_color_material_c()` for a *continuous* scale.

Usage

```
scale_color_material(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_color_material_d(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_color_material_c(palette = "contrast", discrete = FALSE,
  reverse = FALSE, ...)
```

```
scale_colour_material(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_colour_material_c(palette = "contrast", discrete = FALSE,
  reverse = FALSE, ...)

scale_colour_material_d(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_fill_material(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_fill_material_d(palette = "contrast", discrete = TRUE,
  reverse = FALSE, ...)

scale_fill_material_c(palette = "contrast", discrete = FALSE,
  reverse = FALSE, ...)
```

Arguments

palette	Character name of palette. Can be "full", "ice", "rainbow", "complement", or "contrast".
discrete	Boolean indicating whether color aesthetic is discrete or not.
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  theme_modern() +
  scale_fill_material_d()

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_violin() +
  theme_modern() +
  scale_fill_material_d(palette = "ice")

ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Sepal.Length)) +
  geom_point() +
  theme_modern() +
  scale_color_material_c(palette = "rainbow")
```

scale_color_metro *Metro color palette*

Description

The palette based on Metro (<https://www.materialui.co/metrocolors>). Use `scale_color_metro_d` for *discrete* categories and `scale_color_metro_c` for a *continuous* scale.

Usage

```
scale_color_metro(palette = "complement", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_color_metro_d(palette = "complement", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_color_metro_c(palette = "complement", discrete = FALSE,
  reverse = FALSE, ...)
```

```
scale_colour_metro(palette = "complement", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_colour_metro_c(palette = "complement", discrete = FALSE,
  reverse = FALSE, ...)
```

```
scale_colour_metro_d(palette = "complement", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_fill_metro(palette = "complement", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_fill_metro_d(palette = "complement", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_fill_metro_c(palette = "complement", discrete = FALSE,
  reverse = FALSE, ...)
```

Arguments

palette	Character name of palette. Can be "full", "ice", "rainbow", "complement", or "contrast".
discrete	Boolean indicating whether color aesthetic is discrete or not.
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  theme_modern() +
  scale_fill_metro_d()

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_violin() +
  theme_modern() +
  scale_fill_metro_d(palette = "ice")

ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Sepal.Length)) +
  geom_point() +
  theme_modern() +
  scale_color_metro_c(palette = "rainbow")
```

scale_color_pizza *Pizza color palette*

Description

The palette based on authentic neapolitan pizzas. Use `scale_color_pizza_d()` for *discrete* categories and `scale_color_pizza_c()` for a *continuous* scale.

Usage

```
scale_color_pizza(palette = "margherita", discrete = TRUE,
  reverse = FALSE, ...)

scale_color_pizza_d(palette = "margherita", discrete = TRUE,
  reverse = FALSE, ...)

scale_color_pizza_c(palette = "margherita", discrete = FALSE,
  reverse = FALSE, ...)

scale_colour_pizza(palette = "margherita", discrete = TRUE,
  reverse = FALSE, ...)

scale_colour_pizza_c(palette = "margherita", discrete = FALSE,
  reverse = FALSE, ...)

scale_colour_pizza_d(palette = "margherita", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_fill_pizza(palette = "margherita", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_fill_pizza_d(palette = "margherita", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_fill_pizza_c(palette = "margherita", discrete = FALSE,
  reverse = FALSE, ...)
```

Arguments

palette	Pizza type. Can be "margherita" (default), "margherita_crust", "diavola" or "diavola_crust".
discrete	Boolean indicating whether color aesthetic is discrete or not.
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  theme_modern() +
  scale_fill_pizza_d()

ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Sepal.Length)) +
  geom_point() +
  theme_modern() +
  scale_color_pizza_c()
```

scale_color_social *Social color palette*

Description

The palette based on Social (<https://www.materialui.co/socialcolors>). Use `scale_color_social_d` for *discrete* categories and `scale_color_social_c` for a *continuous* scale.

Usage

```
scale_color_social(palette = "complement", discrete = TRUE,
  reverse = FALSE, ...)
```

```
scale_color_social_d(palette = "complement", discrete = TRUE,
```

```

reverse = FALSE, ...)

scale_color_social_c(palette = "complement", discrete = FALSE,
reverse = FALSE, ...)

scale_colour_social(palette = "complement", discrete = TRUE,
reverse = FALSE, ...)

scale_colour_social_c(palette = "complement", discrete = FALSE,
reverse = FALSE, ...)

scale_colour_social_d(palette = "complement", discrete = TRUE,
reverse = FALSE, ...)

scale_fill_social(palette = "complement", discrete = TRUE,
reverse = FALSE, ...)

scale_fill_social_d(palette = "complement", discrete = TRUE,
reverse = FALSE, ...)

scale_fill_social_c(palette = "complement", discrete = FALSE,
reverse = FALSE, ...)

```

Arguments

palette	Character name of palette. Can be "full", "ice", "rainbow", "complement", or "contrast".
discrete	Boolean indicating whether color aesthetic is discrete or not.
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

Examples

```

library(ggplot2)
library(see)

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  theme_modern() +
  scale_fill_social_d()

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_violin() +
  theme_modern() +
  scale_fill_social_d(palette = "ice")

ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Sepal.Length)) +
  geom_point() +
  theme_modern() +

```



```
scale_color_social_c(palette = "rainbow")
```

social_colors

Extract Social colors as hex codes

Description

Can be used to get the hex code of specific colors from the Social color palette. Use `social_colors()` to see all available color.

Usage

```
social_colors(...)
```

Arguments

... Character names of colors.

Value

A character vector with color-codes.

Examples

```
social_colors()
social_colors("dark red", "teal")
```

theme_abyss

Abyss theme

Description

A deep dark blue theme for ggplot.

Usage

```
theme_abyss(base_size = 11, base_family = "", plot.title.size = 15,
  plot.title.face = "plain", plot.title.space = 20,
  legend.position = "right", axis.title.space = 20,
  legend.title.size = 13, legend.text.size = 12,
  axis.title.size = 13, axis.title.face = "plain",
  axis.text.size = 12, axis.text.angle = NULL, tags.size = 15,
  tags.face = "bold")
```

Arguments

<code>base_size</code>	base font size
<code>base_family</code>	base font family
<code>plot.title.size</code>	Title size in pts. Can be "none".
<code>plot.title.face</code>	Title font face ("plain", "italic", "bold", "bold.italic").
<code>plot.title.space</code>	Title spacing.
<code>legend.position</code>	the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)
<code>axis.title.space</code>	Axis title spacing.
<code>legend.title.size</code>	Legend elements text size in pts.
<code>legend.text.size</code>	Legend elements text size in pts. Can be "none".
<code>axis.title.size</code>	Axis title text size in pts.
<code>axis.title.face</code>	Axis font face ("plain", "italic", "bold", "bold.italic").
<code>axis.text.size</code>	Axis text size in pts.
<code>axis.text.angle</code>	Rotate the x axis labels.
<code>tags.size</code>	Tags text size in pts.
<code>tags.face</code>	Tags font face ("plain", "italic", "bold", "bold.italic").

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) +
  geom_point(color = "white") +
  theme_abys()
```

theme_blackboard	<i>Blackboard dark theme</i>
------------------	------------------------------

Description

A modern, sleek and dark theme for ggplot.

Usage

```
theme_blackboard(base_size = 11, base_family = "",
  plot.title.size = 15, plot.title.face = "plain",
  plot.title.space = 20, legend.position = "right",
  axis.title.space = 20, legend.title.size = 13,
  legend.text.size = 12, axis.title.size = 13,
  axis.title.face = "plain", axis.text.size = 12,
  axis.text.angle = NULL, tags.size = 15, tags.face = "bold")
```

Arguments

base_size	base font size
base_family	base font family
plot.title.size	Title size in pts. Can be "none".
plot.title.face	Title font face ("plain", "italic", "bold", "bold.italic").
plot.title.space	Title spacing.
legend.position	the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)
axis.title.space	Axis title spacing.
legend.title.size	Legend elements text size in pts.
legend.text.size	Legend elements text size in pts. Can be "none".
axis.title.size	Axis title text size in pts.
axis.title.face	Axis font face ("plain", "italic", "bold", "bold.italic").
axis.text.size	Axis text size in pts.
axis.text.angle	Rotate the x axis labels.
tags.size	Tags text size in pts.
tags.face	Tags font face ("plain", "italic", "bold", "bold.italic").

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) +
  geom_point(color="white") +
  theme_blackboard()
```

 theme_lucid

Lucid theme

Description

A light, clear theme for ggplot.

Usage

```
theme_lucid(base_size = 11, base_family = "", plot.title.size = 12,
  plot.title.face = "plain", plot.title.space = 15,
  legend.position = "right", axis.title.space = 10,
  legend.title.size = 11, legend.text.size = 10,
  axis.title.size = 11, axis.title.face = "plain",
  axis.text.size = 10, axis.text.angle = NULL, tags.size = 11,
  tags.face = "plain")
```

Arguments

base_size	base font size
base_family	base font family
plot.title.size	Title size in pts. Can be "none".
plot.title.face	Title font face ("plain", "italic", "bold", "bold.italic").
plot.title.space	Title spacing.
legend.position	the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)
axis.title.space	Axis title spacing.
legend.title.size	Legend elements text size in pts.
legend.text.size	Legend elements text size in pts. Can be "none".
axis.title.size	Axis title text size in pts.
axis.title.face	Axis font face ("plain", "italic", "bold", "bold.italic").
axis.text.size	Axis text size in pts.
axis.text.angle	Rotate the x axis labels.
tags.size	Tags text size in pts.
tags.face	Tags font face ("plain", "italic", "bold", "bold.italic").

Examples

```
library(ggplot2)
library(see)

ggplot(iris, aes(x = Sepal.Width, y = Sepal.Length)) +
  geom_point(color = "white") +
  theme_lucid()
```

theme_modern	<i>The easystats' minimal theme</i>
--------------	-------------------------------------

Description

A modern, sleek and elegant theme for ggplot.

Usage

```
theme_modern(base_size = 11, base_family = "", plot.title.size = 15,
  plot.title.face = "plain", plot.title.space = 20,
  legend.position = "right", axis.title.space = 20,
  legend.title.size = 13, legend.text.size = 12,
  axis.title.size = 13, axis.title.face = "plain",
  axis.text.size = 12, axis.text.angle = NULL, tags.size = 15,
  tags.face = "bold")
```

Arguments

base_size	base font size
base_family	base font family
plot.title.size	Title size in pts. Can be "none".
plot.title.face	Title font face ("plain", "italic", "bold", "bold.italic").
plot.title.space	Title spacing.
legend.position	the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)
axis.title.space	Axis title spacing.
legend.title.size	Legend elements text size in pts.
legend.text.size	Legend elements text size in pts. Can be "none".
axis.title.size	Axis title text size in pts.

`axis.title.face` Axis font face ("plain", "italic", "bold", "bold.italic").
`axis.text.size` Axis text size in pts.
`axis.text.angle` Rotate the x axis labels.
`tags.size` Tags text size in pts.
`tags.face` Tags font face ("plain", "italic", "bold", "bold.italic").

Examples

```
library(ggplot2)
library(see)

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

Index

`add_plot_attributes`, 2
`aes()`, 10, 11
`aes_()`, 10, 11

`borders()`, 10, 12

`check_normality`, 5
`colorRampPalette()`, 15–17, 20, 21, 23, 24

`data_plot`, 2, 3
`data_plot()`, 13

`flat_colors`, 7
`fortify()`, 10, 11

`geom_jitter2` (`geom_point2`), 8
`geom_point2`, 8
`geom_pooljitter` (`geom_poolpoint`), 9
`geom_poolpoint`, 9
`geom_violindot`, 10
`geom_violinhalf`, 11
`ggplot()`, 10, 11

`how_to_plot`, 12

`layer()`, 11, 12

`material_colors`, 13
`metro_colors`, 14

`palette_flat`, 14
`palette_material`, 15
`palette_metro`, 15
`palette_pizza`, 16
`palette_social`, 16
`pizza_colors`, 17
`plot()`, 13
`plot.see_bayesfactor_models`
 (`data_plot`), 3
`plot.see_bayesfactor_parameters`
 (`data_plot`), 3

`plot.see_check_distribution`
 (`data_plot`), 3
`plot.see_check_normality` (`data_plot`), 3
`plot.see_check_outliers` (`data_plot`), 3
`plot.see_equivalence_test` (`data_plot`), 3
`plot.see_estimate_density` (`data_plot`), 3
`plot.see_parameters_model` (`data_plot`), 3
`plot.see_parameters_sem` (`data_plot`), 3
`plot.see_point_estimate` (`data_plot`), 3
`plot.see_ropes` (`data_plot`), 3
`plots`, 17

`scale_color_flat`, 18
`scale_color_flat()`, 15
`scale_color_flat_c` (`scale_color_flat`),
 18
`scale_color_flat_d` (`scale_color_flat`),
 18
`scale_color_material`, 19
`scale_color_material()`, 15
`scale_color_material_c`
 (`scale_color_material`), 19
`scale_color_material_d`
 (`scale_color_material`), 19
`scale_color_metro`, 21
`scale_color_metro()`, 16
`scale_color_metro_c`
 (`scale_color_metro`), 21
`scale_color_metro_d`
 (`scale_color_metro`), 21
`scale_color_pizza`, 22
`scale_color_pizza()`, 16
`scale_color_pizza_c`
 (`scale_color_pizza`), 22
`scale_color_pizza_d`
 (`scale_color_pizza`), 22
`scale_color_social`, 23
`scale_color_social()`, 17
`scale_color_social_c`
 (`scale_color_social`), 23

scale_color_social_d
 (scale_color_social), 23

scale_colour_flat (scale_color_flat), 18

scale_colour_flat_c (scale_color_flat),
 18

scale_colour_flat_d (scale_color_flat),
 18

scale_colour_material
 (scale_color_material), 19

scale_colour_material_c
 (scale_color_material), 19

scale_colour_material_d
 (scale_color_material), 19

scale_colour_metro (scale_color_metro),
 21

scale_colour_metro_c
 (scale_color_metro), 21

scale_colour_metro_d
 (scale_color_metro), 21

scale_colour_pizza (scale_color_pizza),
 22

scale_colour_pizza_c
 (scale_color_pizza), 22

scale_colour_pizza_d
 (scale_color_pizza), 22

scale_colour_social
 (scale_color_social), 23

scale_colour_social_c
 (scale_color_social), 23

scale_colour_social_d
 (scale_color_social), 23

scale_fill_flat (scale_color_flat), 18

scale_fill_flat_c (scale_color_flat), 18

scale_fill_flat_d (scale_color_flat), 18

scale_fill_material
 (scale_color_material), 19

scale_fill_material_c
 (scale_color_material), 19

scale_fill_material_d
 (scale_color_material), 19

scale_fill_metro (scale_color_metro), 21

scale_fill_metro_c (scale_color_metro),
 21

scale_fill_metro_d (scale_color_metro),
 21

scale_fill_pizza (scale_color_pizza), 22

scale_fill_pizza_c (scale_color_pizza),
 22

scale_fill_pizza_d (scale_color_pizza),
 22

scale_fill_social (scale_color_social),
 23

scale_fill_social_c
 (scale_color_social), 23

scale_fill_social_d
 (scale_color_social), 23

social_colors, 25

theme_abbyss, 25

theme_blackboard, 26

theme_lucid, 28

theme_modern, 29