Package 'see'

January 15, 2020

Type Package

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

Version 0.4.0

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 (>= 3.2), graphics, grDevices, stats

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

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

RoxygenNote 7.0.2

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 2020-01-15 10:00:02 UTC

R topics documented:

add_plot_attributes . . . 2
coord_radar . . . 3
data_plot . . . 4
flat_colors . . . 11
geom_point2 . . . 12
geom_poolpoint . . . 12
geom_violindot . . . 14
geom_violinhalf . . . 15
material_colors . . . 17
metro_colors . . . 17
palette_flat . . . 18
palette_material . . . 18
palette_metro . . . 19
palette_pizza . . . 19
palette_social . . . 20
pizza_colors . . . 20
plots . . . 21
scale_color_flat . . . 21
scale_color_material . . . 23
scale_color_metro . . . 25
scale_color_pizza . . . 27
scale_color_social . . . 29
social_colors . . . 31
theme_abyss . . . 32
theme_blackboard . . . 33
theme_lucid . . . 34
theme_modern . . . 36
theme_radar . . . 37

Index **40**

`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)
```

---

coord_radar *Radar coordinate system*

Description

Add a radar coordinate system useful for radar charts.

Usage

```
coord_radar(theta = "x", start = 0, direction = 1, ...)
```

Arguments

theta	Can be 'x' or 'y'.
start	Starting position. Best expressed in terms of pi (e.g., -pi/4).
direction	The direction of plotting. Can be 1 or -1.
...	Other arguments to be passed to ggproto.

Examples

```
# Create a radar/spider chart with ggplot:
library(dplyr)
library(tidyr)
library(ggplot2)

data <- iris %>%
  group_by(Species) %>%
  summarise_all(mean) %>%
  pivot_longer(-Species)

data %>%
  ggplot(aes(x = name, y = value, color = Species, group = Species)) +
  geom_polygon(fill = NA, size = 2) +
  coord_radar(start = -pi/4)
```

`data_plot` *Prepare objects for plotting or plot objects*

Description

`data_plot()` attempts to extract and tranform 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"),
  sort = FALSE,
  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_cluster_analysis'
plot(x, data = NULL, n_columns = NULL, size = 0.6, ...)

## S3 method for class 'see_compare_performance'
plot(x, size = 1, ...)

## S3 method for class 'see_equivalence_test'
plot(
  x,
  rope_color = "#0171D3",
  rope_alpha = 0.2,
  show_intercept = FALSE,
  n_columns = 1,
  ...
)

## S3 method for class 'see_estimate_density'
plot(
  x,
  stack = TRUE,
  show_intercept = FALSE,
  n_columns = 1,
  priors = FALSE,
  priors_alpha = 0.4,
  size = 0.9,
  ...
)

## S3 method for class 'see_hdi'
plot(
  x,
  data = NULL,
  show_intercept = FALSE,
  show_zero = TRUE,
  show_title = TRUE,
  n_columns = 1,
  ...
```

```
)

## S3 method for class 'see_n_factors'
plot(x, data = NULL, type = c("bar", "line", "area"), size = 1, ...)

## S3 method for class 'see_p_direction'
plot(
  x,
  data = NULL,
  show_intercept = FALSE,
  priors = FALSE,
  priors_alpha = 0.4,
  n_columns = 1,
  ...
)

## S3 method for class 'see_p_significance'
plot(
  x,
  data = NULL,
  show_intercept = FALSE,
  priors = FALSE,
  priors_alpha = 0.4,
  n_columns = 1,
  ...
)

## S3 method for class 'see_parameters_model'
plot(
  x,
  show_intercept = FALSE,
  point_size = 0.8,
  sort = NULL,
  n_columns = NULL,
  ...
)

## S3 method for class 'see_parameters_pca'
plot(
  x,
  type = c("bar", "line"),
  text_size = 3.5,
  text_color = "black",
  size = 1,
  ...
)

## S3 method for class 'see_parameters_sem'
```

```
plot(
  x,
  data = NULL,
  type = c("regression", "correlation", "loading"),
  threshold_coefficient = NULL,
  threshold_p = NULL,
  ci = TRUE,
  size = 22,
  ...
)

## S3 method for class 'see_parameters_simulate'
plot(
  x,
  data = NULL,
  stack = TRUE,
  show_intercept = FALSE,
  n_columns = NULL,
  ...
)

## 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,
  priors = FALSE,
  priors_alpha = 0.4,
  ...
)

## S3 method for class 'see_rope'
plot(
  x,
  data = NULL,
  rope_alpha = 0.5,
  rope_color = "cadetblue",
  show_intercept = FALSE,
  n_columns = 1,
  ...
)

## S3 method for class 'see_si'
plot(x, si_color = "#0171D3", si_alpha = 0.2, 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.
sort	**Plotting model parameters** If NULL, coefficients are plotted in the order as they appear in the summary. Use sort = "ascending" (or sort = TRUE)) resp. sort = "descending" to sort coefficients in ascending or descending order. **Plotting Bayes factors** Sort pie-slices by posterior probability (descending)?
log	Show log-transformed Bayes factors.
prior_odds	optional vector of prior odds for the models. See BayesFactor::priorOdds. 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, si_color	Color of ROPE/SI ribbon.
rope_alpha, si_alpha	Transparency level of ROPE/SI 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 check_normality, parameters::model_parameters.l or n_factors).
text_size	Size of text labels.
n_columns	For models with multiple components (like fixed and random, count and zero-inflated), defines the number of columns for the panel-layout. If NULL, a single, integrated plot is shown.
size	Size of geoms. Ddepends on the context of the plot() function, so this argument may change size of points, lines or bars.
stack	Logical, if TRUE, densities are plotted as stacked lines. Else, densities are plotted for each parameter among each other.
priors	Logical, if TRUE, prior distributions are simulated (using simulate_prior) and added to the plot.
priors_alpha	Alpha value of the prior distributions.
show_zero	Logical, if TRUE, will add a vertical (dotted) line at 0.
show_title	Logical, if TRUE, will show the title of the plot.
text_color	Color of text labels.

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. *"Mean"*, *"Median"* and/or *"MAP"*) are shown. You may set show_labels = FALSE 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 transparancy 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::p_significance()
- bayestestR::point_estimate()
- bayestestR::rope()
- bayestestR::si()
- modelbased::estimate_contrasts()
- parameters::cluster_analysis()
- parameters::model_parameters()
- parameters::n_factors()
- parameters::parameters_simulate()

- parameters::principal_components()
- 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::compare_performance()
- 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 *Better looking points*

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, n_columns = 2)
```

geom_poolpoint *Pool ball points*

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 *Half-violin Half-dot plot*

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 aes() or aes_(). If specified and inherit.aes = TRUE (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 NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
trim	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, 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? 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.
inherit.aes	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. borders().
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 layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. 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 *Half-violin plot*

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 aes() or aes_(). If specified and inherit.aes = TRUE (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 NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x,10)).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
trim	If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, 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? 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.
inherit.aes	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. borders().
...	Other arguments passed on to layer(). These are often aesthetics, used to set an aesthetic to a fixed value, like colour = "red" or size = 3. They may also be parameters to the paired geom/stat.

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()
```

material_colors *Extract material design colors as hex codes*

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 *Extract Metro colors as hex codes*

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` *Flat UI color palette*

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 `colorRampPalette()`.

Details

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

`palette_material` *Material design color palette*

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 `colorRampPalette()`.

Details

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

---

palette_metro *Metro color palette*

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 `colorRampPalette()`.

Details

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

---

palette_pizza *Pizza color palette*

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 `colorRampPalette()`.

Details

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

palette_social *Social color palette*

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 colorRampPalette().

Details

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

pizza_colors *Extract pizza colors as hex codes*

Description

Extract pizza colors as hex codes

Usage

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

Arguments

...	Character names of pizza ingredients.

Value

A character vector with color-codes.

plots *Multiple plots side by side*

Description

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

Usage

```
plots(..., n_rows = NULL, n_columns = NULL, tags = FALSE)
```

Arguments

...	grobs, gtables, ggplot or trellis objects
n_rows	Number of rows to align plots.
n_columns	Number of columns to align plots.
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, n_columns = 2, tags = TRUE)
plots(p1, p2, n_columns = 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 `discrete_scale()` or `scale_color_gradientn()`, used respectively when discrete 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 colorRampPalette().

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 colorRampPalette().

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 colorRampPalette().

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 colorRampPalette().

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

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_abyss()
```

---

theme_blackboard *Blackboard dark theme*

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` *The easystats' minimal theme*

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()
```

---

theme_radar *Themes for radar plots*

Description

`theme_radar()` is a light, clear theme for ggplot radar-plots, while `theme_radar_dark()` is a dark variant of `theme_radar()`.

Usage

```
theme_radar(
  base_size = 11,
  base_family = "",
  plot.title.size = 12,
  plot.title.face = "plain",
  plot.title.space = 15,
  legend.position = "right",
  axis.title.space = 15,
```

```
  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"
)

theme_radar_dark(
  base_size = 11,
  base_family = "",
  plot.title.size = 12,
  plot.title.face = "plain",
  plot.title.space = 15,
  legend.position = "right",
  axis.title.space = 15,
  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").

See Also

coord_radar

Examples

```
library(ggplot2)
library(dplyr)
library(tidyr)
library(see)

data <- iris %>%
  group_by(Species) %>%
  summarise_all(mean) %>%
  pivot_longer(-Species)

data %>%
  ggplot(aes(
    x = name,
    y = value,
    color = Species,
    group = Species,
    fill = Species
  )) +
  geom_polygon(size = 1, alpha = .1) +
  coord_radar() +
  theme_radar()
```

Index

add_plot_attributes, 2
aes(), *14*, *16*
aes_(), *14*, *16*

borders(), *15, 16*

check_normality, *8*
colorRampPalette(), *18–20*, *24*, *26*, *28*, *30*
coord_radar, 3, *39*

data_plot, *2*, 4

flat_colors, 11
fortify(), *14*, *16*

geom_jitter2 *(*geom_point2*)*, 12
geom_point2, 12
geom_pooljitter *(*geom_poolpoint*)*, 12
geom_poolpoint, 12
geom_violindot, 14
geom_violinhalf, 15
ggplot(), *14*, *16*

layer(), *15, 16*

material_colors, 17
metro_colors, 17

n_factors, *8*

palette_flat, 18
palette_material, 18
palette_metro, 19
palette_pizza, 19
palette_social, 20
pizza_colors, 20
plot.see_bayesfactor_models *(*data_plot*)*, 4
plot.see_bayesfactor_parameters *(*data_plot*)*, 4
plot.see_check_distribution *(*data_plot*)*, 4
plot.see_check_normality *(*data_plot*)*, 4
plot.see_check_outliers *(*data_plot*)*, 4
plot.see_cluster_analysis *(*data_plot*)*, 4
plot.see_compare_performance *(*data_plot*)*, 4
plot.see_equivalence_test *(*data_plot*)*, 4
plot.see_estimate_density *(*data_plot*)*, 4
plot.see_hdi *(*data_plot*)*, 4
plot.see_n_factors *(*data_plot*)*, 4
plot.see_p_direction *(*data_plot*)*, 4
plot.see_p_significance *(*data_plot*)*, 4
plot.see_parameters_model *(*data_plot*)*, 4
plot.see_parameters_pca *(*data_plot*)*, 4
plot.see_parameters_sem *(*data_plot*)*, 4
plot.see_parameters_simulate *(*data_plot*)*, 4
plot.see_point_estimate *(*data_plot*)*, 4
plot.see_rope *(*data_plot*)*, 4
plot.see_si *(*data_plot*)*, 4
plots, 21

scale_color_flat, 21
scale_color_flat(), *18*
scale_color_flat_c *(*scale_color_flat*)*, 21
scale_color_flat_d *(*scale_color_flat*)*, 21
scale_color_material, 23
scale_color_material(), *19*
scale_color_material_c *(*scale_color_material*)*, 23
scale_color_material_d *(*scale_color_material*)*, 23
scale_color_metro, 25
scale_color_metro(), *19*
scale_color_metro_c *(*scale_color_metro*)*, 25
scale_color_metro_d *(*scale_color_metro*)*, 25
scale_color_pizza, 27

scale_color_pizza(), *19*
scale_color_pizza_c
(scale_color_pizza), 27
scale_color_pizza_d
(scale_color_pizza), 27
scale_color_social, 29
scale_color_social(), *20*
scale_color_social_c
(scale_color_social), 29
scale_color_social_d
(scale_color_social), 29
scale_colour_flat (scale_color_flat), 21
scale_colour_flat_c (scale_color_flat), 21
scale_colour_flat_d (scale_color_flat), 21
scale_colour_material
(scale_color_material), 23
scale_colour_material_c
(scale_color_material), 23
scale_colour_material_d
(scale_color_material), 23
scale_colour_metro (scale_color_metro), 25
scale_colour_metro_c
(scale_color_metro), 25
scale_colour_metro_d
(scale_color_metro), 25
scale_colour_pizza (scale_color_pizza), 27
scale_colour_pizza_c
(scale_color_pizza), 27
scale_colour_pizza_d
(scale_color_pizza), 27
scale_colour_social
(scale_color_social), 29
scale_colour_social_c
(scale_color_social), 29
scale_colour_social_d
(scale_color_social), 29
scale_fill_flat (scale_color_flat), 21
scale_fill_flat_c (scale_color_flat), 21
scale_fill_flat_d (scale_color_flat), 21
scale_fill_material
(scale_color_material), 23
scale_fill_material_c
(scale_color_material), 23
scale_fill_material_d
(scale_color_material), 23
scale_fill_metro (scale_color_metro), 25
scale_fill_metro_c (scale_color_metro), 25
scale_fill_metro_d (scale_color_metro), 25
scale_fill_pizza (scale_color_pizza), 27
scale_fill_pizza_c (scale_color_pizza), 27
scale_fill_pizza_d (scale_color_pizza), 27
scale_fill_social (scale_color_social), 29
scale_fill_social_c
(scale_color_social), 29
scale_fill_social_d
(scale_color_social), 29
simulate_prior, *8*
social_colors, 31

theme_abyss, 32
theme_blackboard, 33
theme_lucid, 34
theme_modern, 36
theme_radar, 37
theme_radar_dark (theme_radar), 37