Package ‘see’

June 20, 2022

Type  Package

Title  Model Visualisation Toolbox for ‘easystats’ and ‘ggplot2’

Version  0.7.1

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

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://materialui.co/colors>.

License  GPL-3

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

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

Depends  graphics, grDevices, R (>= 3.5), stats

Imports  bayestestR (>= 0.12.1), datawizard (>= 0.4.1), effectsize (>= 0.7.0), ggplot2 (>= 3.3.6), scales (>= 1.2.0), insight (>= 0.17.1), parameters (>= 0.18.1)

Suggests  brms, coda, correlation, emmeans, factoextra, ggdist, ggraph, ggrepel, gggridges, ggside, glmmTMB, grid, httr, lavaan, lme4, logspline, MASS, mclust, mgcv, metafor, modelbased, NbClust, nFactors, patchwork, performance (>= 0.9.0), poorman, psych, qplotr, randomForest, rmarkdown, rstanarm, splines, testthat (>= 3.1.0), tidygraph, vdiffir

Config/testthat/edition  3

Encoding  UTF-8

Language  en-US

RoxygenNote  7.2.0

NeedsCompilation  no

Author  Daniel Lüdecke [aut, cre] (<https://orcid.org/0000-0002-8895-3206>, @strengjejacke),
Dominique Makowski [aut, inv] (<https://orcid.org/0000-0001-5375-9967>, @Dom_Makowski),
R topics documented:

- add_plot_attributes ........................................... 3
- bluebrown_colors ............................................. 4
- coord_radar ..................................................... 5
- data_plot ....................................................... 6
- flat_colors ..................................................... 8
- geom_binomdensity ............................................ 8
- geom_from_list ............................................... 9
- geom_point2 ................................................... 12
- geom_poolpoint ............................................... 13
- geom_violindot .............................................. 15
- geom_violinhalf ............................................. 16
- golden_ratio .................................................. 18
- material_colors ............................................. 19
- metro_colors ................................................. 19
- palette_bluebrown .......................................... 20
- palette_flat .................................................. 20
- palette_material ............................................ 21
- palette.Metro ............................................... 21
- palette_pizza ................................................ 22
- palette_see ................................................... 22
- palette_social ............................................... 23
- pizza_colors .................................................. 23
- plot.see_bayesfactor_models ............................... 24
- plot.see_bayesfactor_parameters ........................... 25
- plot.see_check_collinearity ................................ 26
- plot.see_check_distribution ................................ 27
- plot.see_check_heteroscedasticity ......................... 27
- plot.see_check_homogeneity ................................ 28
- plot.see_check_normality .................................. 29
- plot.see_check_outliers .................................... 30
- plot.see_compare_parameters ............................... 31
- plot.see_compare_performance .............................. 32
- plot.see_effectsize_table ................................ 33

Repository  CRAN

Date/Publication  2022-06-20 14:20:02 UTC
add_plot_attributes

The `data_plot()` function usually stores information (such as title, axes labels, etc.) as attributes, while `add_plot_attributes()` adds this information to the plot.

**Usage**

```r
add_plot_attributes(x)
```
Arguments

x  An object.

Examples

```r
## 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 <- ggplot(
data,
aes(x = x, y = y, height = height, group = y, fill = fill)
) +
ggridges::geom_ridgeline_gradient()

p + add_plot_attributes(data)

## End(Not run)
```

---

**bluebrown_colors**

*Extract blue-brown colors as hex codes*

Description

Can be used to get the hex code of specific colors from the blue-brown color palette. Use `bluebrown_colors()` to see all available colors.

Usage

`bluebrown_colors(...)`

Arguments

`...` Character names of colors.

Value

A character vector with color-codes.
Examples

bluebrown_colors()

bluebrown_colors("blue", "brown")

---

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  
variable to map angle to (x or y)

start  
Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction.

direction  
1, clockwise; -1, anticlockwise

...  
Other arguments to be passed to ggproto.

Examples

# Create a radar/spider chart with ggplot:
if (require("poorman") && require("ggplot2")) {
  data <- iris[-5] %>%
    aggregate(list(Species = iris$Species), mean) %>%
    datawizard::reshape_longer(2:5)
  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() extracts and transforms an object for plotting, while plot() visualizes results of functions from different packages in easystats-project. See the documentation for your object's class:

- bayestestR::bayesfactor_models()
- bayestestR::bayesfactor_parameters()
- bayestestR::equivalence_test()
- bayestestR::estimate_density()
- bayestestR::hdi()
- bayestestR::p_direction()
- bayestestR::p_significance()
- bayestestR::si()
- effectsize::effectsize()
- modelbased::estimate_contrasts()
- parameters::compare_parameters()
- parameters::describe_distribution()
- parameters::model_parameters()
- parameters::principal_components()
- parameters::n_clusters()
- parameters::n_factors()
- parameters::simulate_parameters()
- performance::check_collinearity()
- performance::check_heteroscedasticity()
- performance::check_homogeneity()
- performance::check_normality()
- performance::check_outliers()
- performance::compare_performance()
- performance::performance_roc()
- performance::check_posterior_predictions()

**Usage**

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

**Arguments**

- **x**
  - An object.
- **data**
  - The original data used to create this object. Can be a statistical model.
- **...**
  - Arguments passed to or from other methods.

**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.

**See Also**

- Package-Vignettes

**Examples**

```r
## Not run:
library(bayestestR)
if (require("rstanarm")) {
  model <- stan_glm(
    Sepal.Length ~ Petal.Width * Species,
    data = iris,
    chains = 2, iter = 200, refresh = 0
  )
  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,
```
```r
refresh = 0,
data = mtcars
)
x <- equivalence_test(model)
plot(x)

## End(Not run)
```

---

**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 colors.

**Usage**

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

**Arguments**

`...`  
Character names of colors.

**Value**

A character vector with color-codes.

**Examples**

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

---

**geom_binomdensity**  
*Add dot-densities for binary y variables*

**Description**

Add dot-densities for binary y variables

**Usage**

```r
geom_binomdensity(data, x, y, scale = "auto", ...)```
Arguments

- **data**: A dataframe.
- **x, y**: Characters corresponding to the x and y axis. Note that y must be a variable with two unique values.
- **scale**: Character specifying method of scaling the dot-densities. Can be: 'auto' (corresponding to the square root of the proportion), 'proportion', 'density' or a custom list with values for each factor level (see examples).

... Other arguments passed to ggdist::geom_dots.

Examples

```r
library(ggplot2)
library(see)

data <- iris[1:100, ]

ggplot() +
  geom_binomdensity(data,
    x = "Sepal.Length",
    y = "Species",
    fill = "red",
    color = NA
  )

# Different scales
data[1:70, "Species"] <- "setosa" # Create unbalanced proportions

ggplot() +
  geom_binomdensity(data, x = "Sepal.Length", y = "Species", scale = "auto")

ggplot() +
  geom_binomdensity(data, x = "Sepal.Length", y = "Species", scale = "density")

ggplot() +
  geom_binomdensity(data, x = "Sepal.Length", y = "Species", scale = "proportion")

ggplot() +
  geom_binomdensity(data,
    x = "Sepal.Length", y = "Species",
    scale = list("setosa" = 0.4, "versicolor" = 0.6)
  )
```

geom_from_list

Create ggplot2 geom(s) from a list

Description

These helper functions are built on top of ggplot2::layer() and can be used to add geom(s), whose type and content are specified as a list.
Usage

geom_from_list(x, ...)  
geoms_from_list(x, ...)

Arguments

x  
A list containing:
  • a geom type (e.g. geom = "point"),
  • a list of aesthetics (e.g. aes = list(x = "mpg", y = "wt")),
  • some data (e.g. data = mtcars),
  • and some other parameters.

For `geoms_from_list()` ("geoms" with an "s"), the input must be a list of lists, ideally named "l1", "l2", "l3", etc.

...  
Additional arguments passed to ggplot2::layer().

Examples

library(ggplot2)

# Example 1 (basic geoms and labels) --------------------------
l1 <- list(
  geom = "point",
data = mtcars,
aes = list(x = "mpg", y = "wt", size = "hp", color = "hp"),
  show.legend = c("size" = FALSE)
)
l2 <- list(
  geom = "labs",
title = "A Title"
)

ggplot() +
geom_from_list(l1) +
geoms_from_list(l2)

ggplot() +
geoms_from_list(list(l1 = l1, l2 = l2))

# Example 2 (Violin, boxplots, ...) --------------------------
l1 <- list(
  geom = "violin",
data = iris,
aes = list(x = "Species", y = "Sepal.Width")
)
l2 <- list(
  geom = "boxplot",
data = iris,
aes = list(x = "Species", y = "Sepal.Width"),
  outlier.shape = NA
```r
l3 <- list(
  geom = "jitter",
  data = iris,
  width = 0.1,
  aes = list(x = "Species", y = "Sepal.Width")
)

ggplot() +
  geom_from_list(l1) +
  geom_from_list(l2) +
  geom_from_list(l3)

# Example 3 (2D density) --------------------------

ggplot() +
  geom_from_list(list(
    geom = "density_2d", data = iris,
    aes = list(x = "Sepal.Width", y = "Petal.Length")
  ))

ggplot() +
  geom_from_list(list(
    geom = "density_2d_filled", data = iris,
    aes = list(x = "Sepal.Width", y = "Petal.Length")
  ))

ggplot() +
  geom_from_list(list(
    geom = "density_2d_polygon", data = iris,
    aes = list(x = "Sepal.Width", y = "Petal.Length")
  )) +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0))

# Example 4 (facet and coord flip) --------------------------

ggplot(iris, aes(x = Sepal.Length, y = Petal.Width)) +
  geom_point() +
  geom_from_list(list(geom = "hline", yintercept = 2)) +
  geom_from_list(list(geom = "coord_flip")) +
  geom_from_list(list(geom = "facet_wrap", facets = "Species", scales = "free"))

# Example 5 (theme and scales) --------------------------

ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_point() +
  geom_from_list(list(geom = "scale_color_viridis_d", option = "inferno")) +
  geom_from_list(list(geom = "theme", legend.position = "top"))

ggplot(iris, aes(x = Sepal.Length, y = Petal.Width, color = Species)) +
  geom_point() +
```
**geom_point2**

Better looking points

**Description**

Somewhat nicer points (especially in case of transparency) without outline strokes (borders, contours) by default.

**Usage**

```r
geom_point2(..., stroke = 0, shape = 16)
geom_jitter2(..., size = 2, stroke = 0, shape = 16)
geom_pointrange2(..., stroke = 0)
geom_count2(..., stroke = 0)
geom_count_borderless(..., stroke = 0)
geom_point_borderless(...)
```
geom_jitter_borderless(...)

geom_pointrange_borderless(...)

Arguments

... Other arguments to be passed to `ggplot2::geom_point()`,
`ggplot2::geom_jitter()`,
`ggplot2::geom_pointrange()`, or `ggplot2::geom_count()`.

stroke Stroke thickness.
shape Shape of points.
size Size of points.

Note

The color aesthetics for `geom_point_borderless()` is "fill", not "color". See 'Examples'.

Examples

```r
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)

ggplot(iris, aes(x = Petal.Width, y = Sepal.Length, fill = Species)) +
  geom_point_borderless(size = 4) +
  theme_modern()

theme_set(theme_abyss())
ggplot(iris, aes(x = Petal.Width, y = Sepal.Length, fill = Species)) +
  geom_point_borderless(size = 4)
```

---

**geom_poolpoint**

**Pool ball points**

Description

Points labelled with the observation name.
Usage

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

```r
gem_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

```r
gemplot2
library(see)

gemplot(iris, aes(x = Petal.Width, y = Sepal.Length, color = Species)) +
gem_poolpoint(label = rownames(iris)) +
scale_color_flat_d() +
theme_modern()
```

```r
gemplot(iris, aes(x = Petal.Width, y = Sepal.Length, color = Species)) +
gem_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 = c("area", "count", "width"),
  show.legend = NA,
  inherit.aes = TRUE,
  dots_size = 0.7,
  dots_color = NULL,
  dots_fill = NULL,
  binwidth = 0.05,
  position_dots = ggplot2::position_nudge(x = -0.025, y = 0),
  ...,  
  size_dots = dots_size,
  color_dots = dots_color,
  fill_dots = dots_fill
)

Arguments

mapping  
Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes()}. If specified and \texttt{inherit.aes} = \texttt{TRUE} (the default), it is combined with the default mapping at the top level of the plot. You must supply \texttt{mapping} if there is no plot mapping.

data  
The data to be displayed in this layer. There are three options:
If \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot()}.
A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{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 \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ head(.x, 10)}).

trim  
If \texttt{TRUE} (default), trim the tails of the violins to the range of the data. If \texttt{FALSE}, don’t trim the tails.
geom_violinhalf

scale
Showing "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().

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_dots = "jitter")
geom_violinhalf

position = "dodge",
trim = TRUE,
flip = FALSE,
scale = c("area", "count", "width"),
show.legend = NA,
inherit.aes = TRUE,
...

Arguments

mapping Set of aesthetic mappings created by \texttt{aes()} or \texttt{aes().} If specified and \texttt{inherit.aes} = \texttt{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 \texttt{NULL}, the default, the data is inherited from the plot data as specified in the call to \texttt{ggplot().}
A \texttt{data.frame}, or other object, will override the plot data. All objects will be fortified to produce a data frame. See \texttt{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 \texttt{data.frame}, and will be used as the layer data. A function can be created from a formula (e.g. \texttt{~ 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 \texttt{TRUE} (default), trim the tails of the violins to the range of the data. If \texttt{FALSE}, don’t trim the tails.
flip Should the half-violin plot switch directions? By default, this is \texttt{FALSE} and all half-violin geoms will have the flat-side on facing leftward. If \texttt{flip = TRUE}, then all flat-sides will face rightward. Optionally, a numeric vector can be supplied indicating which specific geoms should be flipped. See examples for more details.
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? \texttt{NA}, the default, includes if any aesthetics are mapped. \texttt{FALSE} never includes, and \texttt{TRUE} always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes If \texttt{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. \texttt{borders().}

... Other arguments passed on to \texttt{layer().} These are often aesthetics, used to set an aesthetic to a fixed value, like \texttt{colour = "red" or size = 3.} They may also be parameters to the paired geom/stat.
Examples

```r
library(ggplot2)
library(see)

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

# To flip all half-violin geoms, use `flip = TRUE`:
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
   geom_violinhalf(flip = TRUE) +
   theme_modern() +
   scale_fill_material_d()

# To flip the half-violin geoms for the first and third groups only
# by passing a numeric vector
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
   geom_violinhalf(flip = c(1, 3)) +
   theme_modern() +
   scale_fill_material_d()
```

golden_ratio

**Golden Ratio**

**Description**

Returns the golden ratio (1.618034...). Useful to easily obtain golden proportions, for instance for a horizontal figure, if you want its height to be 8, you can set its width to be `golden_ratio(8)`.

**Usage**

```r
golden_ratio(x = 1)
```

**Arguments**

- `x` A number to be multiplied by the golden ratio. The default (x = 1) returns the value of the golden ratio.

**Examples**

```r
golden_ratio()
golden_ratio(10)
```
**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.
### palette_bluebrown

#### Blue-brown design color palette

The palette based on blue-brown colors.

**Usage**

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

**Arguments**

- `palette` Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
- `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_bluebrown()`.

### palette_flat

#### Flat UI color palette

The palette based on Flat UI.

**Usage**

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

**Arguments**

- `palette` Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
- `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.

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

Arguments
- `palette`  Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
- `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.

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

Arguments
- `palette`  Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
- `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

```r
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_see  

See design color palette

Description

See design color palette

Usage

```r
palette_see(palette = "contrast", reverse = FALSE, ...)
```

Arguments

- **palette**: Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
- **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_see()`.
**palette_social**  

*Social color palette*

**Description**

The palette based Social colors.

**Usage**

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

**Arguments**

- `palette`  
  Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).

- `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**

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

**Arguments**

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

**Value**

A character vector with color-codes.
The `plot()` method for the `bayestestR::bayesfactor_models()` function. These plots visualize the posterior probabilities of the compared models.

### Usage

```r
## 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,
...
)
```

### Arguments

- **x**  
  An object.

- **n_pies**  
  Number of pies.

- **value**  
  What value to display.

- **sort**  
  The behavior of this argument depends on the plotting contexts.

  - **Plotting model parameters**: If `NULL`, coefficients are plotted in the order as they appear in the summary. Setting `sort = "ascending"` or `sort = "descending"` sorts coefficients in ascending or descending order, respectively. Setting `sort = TRUE` is the same as `sort = "ascending"`.

  - **Plotting Bayes factors**: Sort pie-slices by posterior probability (descending)?

- **log**  
  Logical that decides whether to display log-transformed Bayes factors.

- **prior_odds**  
  An 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 Bayes Factor), custom `prior_odds` will change the slices’ size.

- **...**  
  Arguments passed to or from other methods.

### Value

A ggplot2-object.
Examples

```r
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", sort = TRUE) +
  scale_fill_pizza(reverse = TRUE)

plot(result, n_pies = "many", value = "BF", log = TRUE) +
  scale_fill_pizza(reverse = FALSE)
```

Description

The `plot()` method for the `bayestestR::bayesfactor_parameters()` function.

Usage

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

Arguments

- **x**: An object.
- **size_point**: Numeric specifying size of point-geoms.
- **rope_color**: Character specifying color of ROPE ribbon.
- **rope_alpha**: Numeric specifying 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.

... Arguments passed to or from other methods.
plot.see_check_collinearity

Plot method for multicollinearity checks

Description

The plot() method for the performance::check_collinearity() function.

Usage

```r
## S3 method for class 'see_check_collinearity'
plot(
  x,
  data = NULL,
  colors = c("#3aaf85", "#1b6ca8", "#cd201f"),
  size_point = 4,
  size_line = 0.8,
  ...
)
```

Arguments

- `x`: An object.
- `data`: The original data used to create this object. Can be a statistical model.
- `colors`: Character vector of length two, indicating the colors (in hex-format) for points and line.
- `size_point`: Numeric specifying size of point-geoms.
- `size_line`: Numeric value specifying size of line geoms.
- `...`: Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

```r
library(performance)
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
result <- check_collinearity(m)
result
plot(result)
```
**plot.see_check_distribution**

*Plot method for classifying the distribution of a model-family*

**Description**

The `plot()` method for the `performance::check_distribution()` function.

**Usage**

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

**Arguments**

- `x`: An object.
- `size_point`: Numeric specifying size of point-geoms.
- `panel`: Logical, if TRUE, plots are arranged as panels; else, single plots are returned.
- `...`: Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

```r
if (require("randomForest") & require("performance")) {
  m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
  result <- check_distribution(m)
  result
  plot(result)
}
```

---

**plot.see_check_heteroscedasticity**

*Plot method for (non-)constant error variance checks*

**Description**

The `plot()` method for the `performance::check_heteroscedasticity()` function.
plot.see_check_homogeneity

Usage

## S3 method for class 'see_check_heteroscedasticity'
plot(x, data = NULL, ...)

Arguments

x      An object.
data    The original data used to create this object. Can be a statistical model.
...     Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

library(performance)
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
result <- check_heteroscedasticity(m)
result
plot(result, data = m)  # data required for pkgdown
Examples

```r
library(performance)
model <- lm(len ~ supp + dose, data = ToothGrowth)
result <- check_homogeneity(model)
result
plot(result)
```

Description

The `plot()` method for the `performance::check_normality()` function.

Usage

```r
## S3 method for class 'see_check_normality'
plot(x, 
     type = c("density", "qq", "pp"),
     data = NULL,
     size_line = 0.8,
     size_point = 2,
     alpha = 0.2,
     dot_alpha = 0.8,
     colors = c("#3aaf85", "#1b6ca8"),
     detrend = FALSE,
     ...)
```

Arguments

- `x` An object.
- `type` Character vector, indicating the type of plot.
- `data` The original data used to create this object. Can be a statistical model.
- `size_line` Numeric value specifying size of line geoms.
- `size_point` Numeric specifying size of point-geoms.
- `alpha` Numeric value specifying alpha level of the confidence bands.
- `dot_alpha` Numeric value specifying alpha level of the point geoms.
- `colors` Character vector of length two, indicating the colors (in hex-format) for points and line.
- `detrend` Logical that decides if the plot should be detrended.
- `...` Arguments passed to or from other methods.
plot.see_check_outliers

Plot method for checking outliers

Value

A ggplot2-object.

Examples

library(performance)
m <<- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
result <- check_normality(m)
plot(result)

plot.see_check_outliers

Description

The plot() method for the performance::check_outliers() function.

Usage

## S3 method for class 'see_check_outliers'
plot(
x,
size_text = 3.5,
size_line = 0.8,
dot_alpha = 0.8,
colors = c("#3aaf85", "#1b6ca8", "#cd201f"),
rescale_distance = TRUE,
type = c("dots", "bars"),
show_labels = TRUE,
...
)

Arguments

x An object.
size_text Numeric value specifying size of text labels.
size_line Numeric value specifying size of line geoms.
dot_alpha Numeric value specifying alpha level of the point geoms.
colors Character vector of length two, indicating the colors (in hex-format) for points and line.
rescale_distance Logical. If TRUE, distance values are rescaled to a range from 0 to 1. This is mainly due to better catch the differences between distance values.
type Character vector, indicating the type of plot.
show_labels Logical. If TRUE, text labels are displayed.
... Arguments passed to or from other methods.
plot.see_compare_parameters

Value

A ggplot2-object.

Examples

```r
library(performance)
data(mtcars)
mt1 <- mtcars[, c(1, 3, 4)]
mt2 <- rbind(
  mt1,
  data.frame(mpg = c(37, 40), disp = c(300, 400), hp = c(110, 120))
)
model <- lm(disp ~ mpg + hp, data = mt2)
plot(check_outliers(model))
```

plot.see_compare_parameters

Plot method for comparison of model parameters

Description

The `plot()` method for the `parameters::compare_parameters()` function.

Usage

```r
## S3 method for class 'see_compare_parameters'
plot(
  x,
  show_intercept = FALSE,
  size_point = 0.8,
  size_text = NA,
  dodge_position = 0.8,
  sort = NULL,
  n_columns = NULL,
  show_labels = FALSE,
  ...
)
```

Arguments

- `x` An object.
- `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.
- `size_point` Numeric specifying size of point-geoms.
- `size_text` Numeric value specifying size of text labels.
dodge_position Numeric value specifying the amount of "dodging" (spacing) between geoms.
sort The behavior of this argument depends on the plotting contexts.
  • Plotting model parameters: If NULL, coefficients are plotted in the order as they appear in the summary. Setting sort = "ascending" or sort = "descending" sorts coefficients in ascending or descending order, respectively. Setting sort = TRUE is the same as sort = "ascending".
  • Plotting Bayes factors: Sort pie-slices by posterior probability (descending)?
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.
show_labels Logical. If TRUE, text labels are displayed.
...
Arguments passed to or from other methods.

Value
A ggplot2-object.

Examples
if (require("insight") && require("parameters") && packageVersion("insight") >= "0.13.0") {
data(iris)
lm1 <- lm(Sepal.Length ~ Species, data = iris)
lm2 <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
lm3 <- lm(Sepal.Length ~ Species * Petal.Length, data = iris)
result <- compare_parameters(lm1, lm2, lm3)
plot(result)
}

plot.see_compare_performance

Plot method for comparing model performances

Description
The plot() method for the performance::compare_performance() function.

Usage
## S3 method for class 'see_compare_performance'
plot(x, size_line = 1, ...)
plot.see_effectsize_table

Arguments

- x: An object.
- size_line: Numeric value specifying size of line geoms.
- ...: Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

```r
library(performance)
data(iris)
lm1 <- lm(Sepal.Length ~ Species, data = iris)
lm2 <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
lm3 <- lm(Sepal.Length ~ Species * Petal.Length, data = iris)
result <- compare_performance(lm1, lm2, lm3)
result
plot(result)
```

plot.see_effectsize_table

*Plot method for effect size tables*

Description

The plot() method for the effectsize::effectsize() function.

Usage

```r
## S3 method for class 'see_effectsize_table'
plot(x, ...)
```

Arguments

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

Value

A ggplot2-object.

Examples

```r
library(effectsize)
m <- aov(mpg ~ factor(am) * factor(cyl), data = mtcars)
result <- eta_squared(m)
plot(result)
```
### Description

The `plot()` method for the `bayestestR::equivalence_test()` function.

### Usage

```r
## S3 method for class 'see_equivalence_test_effectsize'
plot(x, ...)

## 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_equivalence_test_lm'
plot(
    x,
    size_point = 0.7,
    rope_color = "#0171D3",
    rope_alpha = 0.2,
    show_intercept = FALSE,
    n_columns = 1,
    ...
)
```

### Arguments

- **x**: An object.
- **...**: Arguments passed to or from other methods.
- **rope_color**: Character specifying color of ROPE ribbon.
- **rope_alpha**: Numeric specifying 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.
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_point Numeric specifying size of point-geoms.

Value

A ggplot2-object.

Examples

library(effectsize)
m <- aov(mpg ~ factor(am) * factor(cyl), data = mtcars)
result <- eta_squared(m)
plot(result)

plot.see_estimate_contrasts

Plot method for estimating contrasts

Description

The plot() method for the modelbased::estimate_contrasts() function.

Usage

## S3 method for class 'see_estimate_contrasts'
plot(x, data = NULL, ...)

Arguments

x An object.
data The original data used to create this object. Can be a statistical model.
...

Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

if (require("modelbased") && require("rstanarm")) {
  model <- stan_glm(Sepal.Width ~ Species, data = iris, refresh = 0)
  contrasts <- estimate_contrasts(model)
  means <- estimate_means(model)
  plot(contrasts, means)
}
plot.see_estimate_density

Plot method for density estimation of posterior samples

Description

The plot() method for the bayestestR::estimate_density() function.

Usage

```r
## S3 method for class 'see_estimate_density'
plot(
  x,
  stack = TRUE,
  show_intercept = FALSE,
  n_columns = 1,
  priors = FALSE,
  priors_alpha = 0.4,
  posteriors_alpha = 0.7,
  size_line = 0.9,
  size_point = 2,
  centrality = "median",
  ci = 0.95,
  ...
)
```

Arguments

- **x** An object.
- **stack** Logical. If TRUE, densities are plotted as stacked lines. Else, densities are plotted for each parameter among each other.
- **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.
- **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.
- **priors** Logical. If TRUE, prior distributions are simulated (using bayestestR::simulate_prior()) and added to the plot.
- **priors_alpha** Numeric value specifying alpha for the prior distributions.
- **posteriors_alpha** Numeric value specifying alpha for the posterior distributions.
- **size_line** Numeric value specifying size of line geoms.
- **size_point** Numeric specifying size of point-geoms.
plot.see_hdi

Character specifying the point-estimate (centrality index) to compute. Can be "median", "mean" or "MAP".

Numeric value of probability of the CI (between 0 and 1) to be estimated. Default to .95.

Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

if (require("bayestestR") && require("rstanarm")) {
  set.seed(123)
  m <- stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0)
  result <- estimate_density(m)
  plot(result)
}

plot.see_hdi

Plot method for uncertainty or credible intervals

Description

The plot() method for the bayestestR::hdi() and related function.

Usage

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

Arguments

x
  An object.

data
  The original data used to create this object. Can be a statistical model.
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.

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.

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.

... Arguments passed to or from other methods.

Value
A ggplot2-object.

Examples

if (require("bayestestR") & & require("rstanarm")) {
  set.seed(123)
  m <- stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0)
  result <- hdi(m)
  result
  plot(result)
}

plot.see_n_factors Plot method for numbers of clusters to extract or factors to retain

Description
The plot() method for the parameters::n_factors() and parameters::n_clusters()

Usage

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

Arguments

x An object.

data The original data used to create this object. Can be a statistical model.

type Character vector, indicating the type of plot.

size Depending on type, a numeric value specifying size of bars, lines, or segments.

... Arguments passed to or from other methods.
Value

A ggplot2-object.

Examples

```r
if (require("parameters") && require("nFactors")) {
  data(mtcars)
  result <- n_factors(mtcars, type = "PCA")
  result
  plot(result, type = "line")
}
```

Description

The `plot()` method for the `parameters::model_parameters()` function when used with brms-meta-analysis models.

Usage

```r
## S3 method for class 'see_parameters_brms_meta'
plot(
  x, 
  size_point = 2, 
  size_line = 0.8, 
  size_text = 3.5, 
  posteriors_alpha = 0.7, 
  rope_alpha = 0.15, 
  rope_color = "cadetblue", 
  normalize_height = TRUE, 
  show_labels = TRUE, 
  ...
)
```

Arguments

- `x` An object.
- `size_point` Numeric specifying size of point-geoms.
- `size_line` Numeric value specifying size of line geoms.
- `size_text` Numeric value specifying size of text labels.
- `posteriors_alpha` Numeric value specifying alpha for the posterior distributions.
- `rope_alpha` Numeric specifying transparency level of ROPE ribbon.
rope_color  Character specifying color of ROPE ribbon.
normalize_height  Logical. If TRUE, height of mcmc-areas is "normalized", to avoid overlap. In certain cases when the range of a posterior distribution is narrow for some parameters, this may result in very flat mcmc-areas. In such cases, set normalize_height = FALSE.
show_labels  Logical. If TRUE, text labels are displayed.
...
Arguments passed to or from other methods.

Details

**Colors of density areas and errorbars:** To change the colors of the density areas, use `scale_fill_manual()` with named color-values, e.g. `scale_fill_manual(values = c("Study" = "blue", "Overall" = "green"))`. To change the color of the error bars, use `scale_color_manual(values = c("Errorbar" = "red"))`.

**Show or hide estimates and CI:** Use `show_labels = FALSE` to hide the textual output of estimates and credible intervals.

Value

A ggplot2-object.

Examples

```r
## Not run:
if (require("bayestestR") && require("brms") && require("metafor")) {
  
  # data
data(dat.bcg)
dat <- escalc(
    measure = "RR",
    ai = tpos,
    bi = tneg,
    ci = cpos,
    di = cneg,
    data = dat.bcg
  )
dat$author <- make.unique(dat$author)

  # model
set.seed(123)
priors <- c(
  prior(normal(0, 1), class = Intercept),
  prior(cauchy(0, 0.5), class = sd)
)
model <- brm(yi | se(vi) ~ 1 + (1 | author), data = dat)

  # result
mp <- model_parameters(model)
plot(mp)
```
plot.see_parameters_distribution

## S3 method for class 'see_parameters_distribution'
plot(
  x,
  dispersion = FALSE,
  dispersion_alpha = 0.3,
  dispersion_color = "#3498db",
  dispersion_style = c("ribbon", "curve"),
  size_bar = 0.7,
  highlight = NULL,
  highlight_color = NULL,
  ...
)

Arguments

x An object.
dispersion Logical. If TRUE, a range of dispersion for each variable to the plot will be added.
dispersion_alpha Numeric value specifying the transparency level of dispersion ribbon.
dispersion_color Character specifying the color of dispersion ribbon.
dispersion_style Character describing the style of dispersion area. "ribbon" for a ribbon, "curve" for a normal-curve.
size_bar Size of bar geoms.
highlight A vector with names of categories in x that should be highlighted.
highlight_color A vector of color values for highlighted categories. The remaining (non-highlighted) categories will be filled with a lighter grey.
... Arguments passed to or from other methods.
Value

A ggplot2-object.

Examples

```r
library(parameters)
set.seed(333)
x <- sample(1:100, 1000, replace = TRUE)
result <- describe_distribution(x)
result
plot(result)
```

Plot method for model parameters

The plot() method for the parameters::model_parameters() function.

Usage

```r
## S3 method for class 'see_parameters_model'
plot(x,
     show_intercept = FALSE,
     size_point = 0.8,
     size_text = NA,
     sort = NULL,
     n_columns = NULL,
     type = c("forest", "funnel"),
     weight_points = TRUE,
     show_labels = FALSE,
     show_estimate = TRUE,
     show_interval = TRUE,
     show_density = FALSE,
     log_scale = FALSE,
     ...
)
```

```r
## S3 method for class 'see_parameters_sem'
plot(x,
     data = NULL,
     component = c("regression", "correlation", "loading"),
     type = component,
     threshold_coefficient = NULL,
...}
```
threshold_p = NULL,
  ci = TRUE,
  size_point = 22,
  ...)
}

Arguments

x An object.

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.

size_point Numeric specifying size of point-geoms.

size_text Numeric value specifying size of text labels.

sort The behavior of this argument depends on the plotting contexts.

  • Plotting model parameters: If NULL, coefficients are plotted in the order as they appear in the summary. Setting sort = "ascending" or sort = "descending" sorts coefficients in ascending or descending order, respectively. Setting sort = TRUE is the same as sort = "ascending".
  
  • Plotting Bayes factors: Sort pie-slices by posterior probability (descending)?

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.

type Character indicating the type of plot. Only applies for model parameters from meta-analysis objects (e.g. `metafor`).

weight_points Logical. If TRUE, for meta-analysis objects, point size will be adjusted according to the study-weights.

show_labels Logical. If TRUE, text labels are displayed.

show_estimate Should the point estimate of each parameter be shown? (default: TRUE)

show_interval Should the compatibility interval(s) of each parameter be shown? (default: TRUE)

show_density Should the compatibility density (i.e., posterior, bootstrap, or confidence density) of each parameter be shown? (default: FALSE)

log_scale Should exponentiated coefficients (e.g., odds-ratios) be plotted on a log scale? (default: FALSE)

... Arguments passed to or from other methods.

data The original data used to create this object. Can be a statistical model.

component Character indicating which component of the model should be plotted.

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.
Value

A ggplot2-object.

Examples

```r
library(parameters)
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
result <- model_parameters(m)
result
plot(result)
```

---

### plot.see_parameters_pca

**Plot method for principal component analysis**

**Description**

The `plot()` method for the `parameters::principal_components()` function.

**Usage**

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

**Arguments**

- **x** An object.
- **type** Character vector, indicating the type of plot.
- **size_text** Numeric value specifying size of text labels.
- **text_color** Character specifying color of text labels.
- **size** Depending on type, a numeric value specifying size of bars, lines, or segments.
- **show_labels** Logical. If TRUE, text labels are displayed.
- **...** Arguments passed to or from other methods.

**Value**

A ggplot2-object.
Examples

```r
library(parameters)
data(mtcars)
result <- principal_components(mtcars[, 1:7], n = "all", threshold = 0.2)
result
plot(result)
```

---

**plot.see_parameters_simulate**  
*Plot method for simulated model parameters*

**Description**

The `plot()` method for the `parameters::simulate_parameters()` function.

**Usage**

```r
## S3 method for class 'see_parameters_simulate'
plot(
  x,
  data = NULL,
  stack = TRUE,
  show_intercept = FALSE,
  n_columns = NULL,
  normalize_height = FALSE,
  size_line = 0.9,
  posteriors_alpha = 0.7,
  centrality = "median",
  ci = 0.95,
  ...
)
```

**Arguments**

- **x**  
  An object.
- **data**  
  The original data used to create this object. Can be a statistical model.
- **stack**  
  Logical. If TRUE, densities are plotted as stacked lines. Else, densities are plotted for each parameter among each other.
- **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.
- **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.
normalize_height

Logical. If TRUE, height of density-areas is "normalized", to avoid overlap. In certain cases when the range of a distribution of simulated draws is narrow for some parameters, this may result in very flat density-areas. In such cases, set normalize_height = FALSE.

size_line

Numeric value specifying size of line geoms.

posteriors_alpha

Numeric value specifying alpha for the posterior distributions.

centrality

Character specifying the point-estimate (centrality index) to compute. Can be "median", "mean" or "MAP".

... Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

library(parameters)
m <- lm(mpg ~ wt + cyl + gear, data = mtcars)
result <- simulate_parameters(m)
result
plot(result)

plot.see_performance_roc

Plot method for ROC curves

Description

The plot() method for the performance::performance_roc() function.

Usage

## S3 method for class 'see_performance_roc'
plot(x, ...)

Arguments

x An object.

... Arguments passed to or from other methods.

Value

A ggplot2-object.
Examples

```r
library(performance)
data(iris)
set.seed(123)
iris$y <- rbinom(nrow(iris), size = 1, .3)
folds <- sample(nrow(iris), size = nrow(iris) / 8, replace = FALSE)
test_data <- iris[folds, ]
train_data <- iris[-folds, ]

model <- glm(y ~ Sepal.Length + Sepal.Width, data = train_data, family = "binomial")
result <- performance_roc(model, new_data = test_data)
result
plot(result)
```

----

plot.see_point_estimate

Plot method for point estimates of posterior samples

Description

The `plot()` method for the `bayestestR::point_estimate()`.

Usage

```r
## S3 method for class 'see_point_estimate'
plot(
  x,
  data = NULL,
  size_point = 2,
  size_text = 3.5,
  panel = TRUE,
  show_labels = TRUE,
  show_intercept = FALSE,
  priors = FALSE,
  priors_alpha = 0.4,
  ...
)
```

Arguments

- `x` An object.
- `data` The original data used to create this object. Can be a statistical model.
- `size_point` Numeric specifying size of point-geoms.
- `size_text` Numeric value specifying size of text labels.
- `panel` Logical, if TRUE, plots are arranged as panels; else, single plots are returned.
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.

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.

priors Logical. If TRUE, prior distributions are simulated (using `bayestestR::simulate_prior()`) and added to the plot.

priors_alpha Numeric value specifying alpha for the prior distributions.

... Arguments passed to or from other methods.

Value
A ggplot2-object.

Examples

```r
if (require("bayestestR") && require("rstanarm")) {
  set.seed(123)
  m <- stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0)
  result <- point_estimate(m, centrality = "median")
  plot(result)
}
```

---

plot.see_p_direction  Plot method for probability of direction

Description
The plot() method for the `bayestestR::p_direction()` function.

Usage

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

x
An object.
data
The original data used to create this object. Can be a statistical model.
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.
priors
Logical. If TRUE, prior distributions are simulated (using bayestestR::simulate_prior()) and added to the plot.
priors_alpha
Numeric value specifying alpha for the prior distributions.
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.
...
Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

if (require("bayestestR") && require("rstanarm")) {
  set.seed(123)
  m <<- stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0)
  result <- p_direction(m)
  plot(result)
}

plot.see_p_significance

Plot method for practical significance

Description

The plot() method for the bayestestR::p_significance() function.

Usage

## S3 method for class 'see_p_significance'
plot(
  x,
  data = NULL,
  show_intercept = FALSE,
  priors = FALSE,
Arguments

- **x**: An object.
- **data**: The original data used to create this object. Can be a statistical model.
- **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.
- **priors**: Logical. If TRUE, prior distributions are simulated (using `bayestestR::simulate_prior()`) and added to the plot.
- **priors_alpha**: Numeric value specifying alpha for the prior distributions.
- **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.
- **...**: Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

```r
if (require("bayestestR") && require("rstanarm")) {
  set.seed(123)
  m <- stan_glmer(Sepal.Length ~ Petal.Width + Species, data = iris, refresh = 0)
  result <- p_significance(m)
  plot(result)
}
```

---

**plot.see_rope**

**Plot method for Region of Practical Equivalence**

**Description**

The plot() method for the `bayestestR::rope()`.
Usage

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

Arguments

x  
An object.

data  
The original data used to create this object. Can be a statistical model.

rope_alpha  
Numeric specifying transparency level of ROPE ribbon.

rope_color  
Character specifying color 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.

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.

...  
Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

if (require("bayestestR") && require("rstanarm")) {
  set.seed(123)
  m <- stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0)
  result <- rope(m)
  result
  plot(result)
}
plot.see_si  Plot method for support intervals

Description

The plot() method for the bayestestR::si().

Usage

## S3 method for class 'see_si'
plot(
  x,
  si_color = "#0171D3",
  si_alpha = 0.2,
  show_intercept = FALSE,
  support_only = FALSE,
  ...
)

Arguments

x  An object.

si_color  Character specifying color of SI ribbon.

si_alpha  Numeric value specifying Transparency level of 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.

support_only  Logical. Decides whether to plot only the support data, or show the "raw" prior and posterior distributions? Only applies when plotting bayestestR::si().

...  Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

if (require("bayestestR") && require("rstanarm")) {
  set.seed(123)
  m <- stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0)
  result <- si(m)
  result
  plot(result)
}
A wrapper around patchwork to plot multiple figures side by side on the same page. See the patchwork documentation for more advanced control of plot layouts.

Usage

plots(
  ..., 
  n_rows = NULL, 
  n_columns = NULL, 
  guides = NULL, 
  tags = FALSE, 
  tag_prefix = NULL, 
  tag_suffix = NULL, 
  tag_sep = NULL, 
  title = NULL, 
  subtitle = NULL, 
  caption = NULL, 
  theme = NULL
)

Arguments

... Multiplot ggplots or a list containing ggplot objects
n_rows Number of rows to align plots.
n_columns Number of columns to align plots.
guides A string specifying how guides should be treated in the layout. 'collect' will collect shared guides across plots, removing duplicates. 'keep' will keep guides alongside their plot. 'auto' will inherit from a higher patchwork level (if any). See patchwork::plot_layout() for details.
tags Add tags to your subfigures. Can be NULL to omit (default) or a character vector containing tags for each plot. Automatic tags can also be generated with '1' for Arabic numerals, 'A' for uppercase Latin letters, 'a' for lowercase Latin letters, 'I' for uppercase Roman numerals, and 'i' for lowercase Roman numerals. For backwards compatibility, can also be FALSE (equivalent to NULL), NA (equivalent to NULL), or TRUE (equivalent to 'A').
tag_prefix, tag_suffix Text strings that should appear before or after the tag.
tag_sep Text string giving the separator to use between different tag levels.
title, subtitle, caption Text strings to use for the various plot annotations to add to the composed patchwork.
theme

A ggplot theme specification to use for the plot. Only elements related to titles, caption, and tags, as well as plot margin and background, are used.

Examples

```r
library(ggplot2)
library(see)

p1 <- ggplot(mtcars, aes(x = disp, y = mpg)) + geom_point()
p2 <- ggplot(mtcars, aes(x = mpg)) + geom_density()
p3 <- ggplot(mtcars, aes(x = factor(cyl))) + geom_bar() + scale_x_discrete("cyl")

plots(p1, p2)
plots(p1, p2, n_columns = 2, tags = "A")
plots(p1, p2, n_columns = 1, tags = c("Fig. 1", "Fig. 2", "Fig. 3"),
      title = "The surprising truth about mtcars")
```

print.see_performance_pp_check

Plot method for posterior predictive checks

Description

The `plot()` method for the `performance::check_predictions()` function.

Usage

```r
## S3 method for class 'see_performance_pp_check'
print(
  x,
  size_line = 0.5,
  line_alpha = 0.15,
  size_bar = 0.7,
  style = theme_lucid,
  colors = unname(social_colors(c("green", "blue"))),
  ...
)

## S3 method for class 'see_performance_pp_check'
plot(
  x,
  size_line = 0.5,
```
Arguments

- `x`  An object.
- `size_line`  Numeric value specifying size of line geoms.
- `line_alpha`  Numeric value specifying alpha of lines indicating yrep.
- `size_bar`  Size of bar geoms.
- `style`  A ggplot2-theme.
- `colors`  Character vector of length two, indicating the colors (in hex-format) for points and line.
- `...`  Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

```r
if (require("performance")) {
  model <- lm(Sepal.Length ~ Species * Petal.Width + Petal.Length, data = iris)
  check_posterior_predictions(model)
}
```

scale_color_bluebrown  

Blue-brown color palette

Description

A blue-brown color palette. Use `scale_color_bluebrown_d()` for discrete categories and `scale_color_bluebrown_c()` for a continuous scale.

Usage

```r
scale_color_bluebrown(
  palette = "contrast",
  discrete = TRUE,
  reverse = FALSE,
  ...
)
```
scale_color_bluebrown_d(
    palette = "contrast",
    discrete = TRUE,
    reverse = FALSE,
    ...
)

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

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

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

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

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

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

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

Arguments

palette        Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
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_bluebrown_d()

scale_color_flat      Flat UI color palette

Description

The palette based on Flat UI. 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. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
- **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

```r
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. Use `scale_color_material_d()` for *discrete* categories and `scale_color_material_c()` for a *continuous* scale.

**Usage**

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

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

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

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

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

```r
scale_colour_material_d(
  palette = "contrast",
```
scale_color_material

```r
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. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
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")
```
scale_color_metro

```
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 Metro colors. 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_color_metro

... 

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. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
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")
scale_color_pizza

```
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

```r
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_see

See color palette

Description

The See color palette. Use scale_color_see_d() for discrete categories and scale_color_see_c() for a continuous scale.

Usage

scale_color_see(palette = "contrast", discrete = TRUE, reverse = FALSE, ...)
scale_color_see_d(palette = "contrast", discrete = TRUE, reverse = FALSE, ...)
scale_color_see_c(palette = "contrast", discrete = FALSE, reverse = FALSE, ...)
scale_colour_see(palette = "contrast", discrete = TRUE, reverse = FALSE, ...)

scale_colour_see_c(
  palette = "contrast",
  discrete = FALSE,
  reverse = FALSE,
  ...
)
scale_colour_see_d(palette = "contrast", discrete = TRUE, reverse = FALSE, ...)
scale_fill_see(palette = "contrast", discrete = TRUE, reverse = FALSE, ...)
scale_fill_see_d(palette = "contrast", discrete = TRUE, reverse = FALSE, ...)
scale_fill_see_c(palette = "contrast", discrete = FALSE, reverse = FALSE, ...)

Arguments

- **palette**: Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
- **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_see_d()

ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width, colour = Species)) +
  geom_point() +
  theme_abyss() +
  scale_colour_see(palette = "light")

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

---

scale_color_social   Social color palette

Description

The palette based Social colors. Use scale_color_social_d for discrete categories and scale_color_social_c for a continuous scale.

Usage

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

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

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

```r
scale_colour_social(
  palette = "complement",
```
scale_color_social

  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. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
  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().
**see_colors**

Examples

```r
library(ggplot2)
library(see)

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

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

# Scatter plot with color
ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Sepal.Length)) +
  geom_point() +
  theme_modern() +
  scale_color_social_c(palette = "rainbow")
```

---

**see_colors**

Extract See colors as hex codes

Description

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

Usage

```r
see_colors(...)```

Arguments

... Character names of colors.

Value

A character vector with color-codes.

Examples

```r
see_colors()

see_colors("indigo", "lime")```
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,
  plot.title.position = "plot",
  legend.position = "right",
  axis.title.space = 20,
  legend.title.size = 13,
  legend.text.size = 12,
)
theme_abyss

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, given in pts.
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.
plot.title.position Alignment of the plot title/subtitle and caption. The setting for plot.title.position applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).
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,
  plot.title.position = "plot",
  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, given in pts.
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.
theme_lucid

plot.title.position
Alignment of the plot title/subtitle and caption. The setting for plot.title.position applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).

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,
    plot.title.position = "plot",
    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, given in pts.
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.
plot.title.position  Alignment of the plot title/subtitle and caption. The setting for plot.title.position applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).
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,
  plot.title.position = "plot",
  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"
)
**theme_modern**

Arguments

- **base_size**: base font size, given in pts.
- **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.
- **plot.title.position**: Alignment of the plot title/subtitle and caption. The setting for plot.title.position applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).
- **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

```r
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

```r
theme_radar(
  base_size = 11,
  base_family = "",  
  plot.title.size = 12, 
  plot.title.face = "plain", 
  plot.title.space = 15, 
  plot.title.position = "plot", 
  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"
)
```

```r
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, given in pts.
- **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.
- **plot.title.position**: Alignment of the plot title/subtitle and caption. The setting for plot.title.position applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).
- **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

```r
if (require("ggplot2") && require("poorman")) {
  data <- iris[-5] %>%
    aggregate(list(Species = iris$Species), mean) %>%
    datawizard::reshape_longer(2:5)
}
data %>%
ggplot(aes(
  x = Name,
  y = Value,
  color = Species,
  group = Species,
  fill = Species
)) +
geom_polygon(size = 1, alpha = .1) +
coord_radar() +
theme_radar()
## Index

<table>
<thead>
<tr>
<th>Function / Package</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_plot_attributes</td>
<td>3</td>
</tr>
<tr>
<td>aes()</td>
<td>15, 17</td>
</tr>
<tr>
<td>aes_()</td>
<td>15, 17</td>
</tr>
<tr>
<td>bayestestR::bayesfactor_models()</td>
<td>6</td>
</tr>
<tr>
<td>bayestestR::bayesfactor_parameters()</td>
<td>6</td>
</tr>
<tr>
<td>bayestestR::equivalence_test()</td>
<td>6</td>
</tr>
<tr>
<td>bayestestR::estimate_density()</td>
<td>6</td>
</tr>
<tr>
<td>bayestestR::hdi()</td>
<td>6</td>
</tr>
<tr>
<td>bayestestR::p_direction()</td>
<td>6</td>
</tr>
<tr>
<td>bayestestR::p_significance()</td>
<td>6</td>
</tr>
<tr>
<td>bayestestR::si()</td>
<td>6, 52</td>
</tr>
<tr>
<td>bayestestR::simulate_prior()</td>
<td>36, 48–50</td>
</tr>
<tr>
<td>bluebrown_colors</td>
<td>4</td>
</tr>
<tr>
<td>borders()</td>
<td>16, 17</td>
</tr>
<tr>
<td>colorRampPalette()</td>
<td>20–23, 57, 60, 62, 64, 65, 67</td>
</tr>
<tr>
<td>coord_radar</td>
<td>5</td>
</tr>
<tr>
<td>coord_radar()</td>
<td>77</td>
</tr>
<tr>
<td>data_plot</td>
<td>6</td>
</tr>
<tr>
<td>effectsize::effectsize()</td>
<td>6</td>
</tr>
<tr>
<td>flat_colors</td>
<td>8</td>
</tr>
<tr>
<td>fortify()</td>
<td>15, 17</td>
</tr>
<tr>
<td>geom_binomdensity</td>
<td>8</td>
</tr>
<tr>
<td>geom_count2 (geom_point2)</td>
<td>12</td>
</tr>
<tr>
<td>geom_count_borderless (geom_point2)</td>
<td>12</td>
</tr>
<tr>
<td>geom_from_list</td>
<td>9</td>
</tr>
<tr>
<td>geom_jitter2 (geom_point2)</td>
<td>12</td>
</tr>
<tr>
<td>geom_jitter_borderless (geom_point2)</td>
<td>12</td>
</tr>
<tr>
<td>geom_point2</td>
<td>12</td>
</tr>
<tr>
<td>geom_point_borderless (geom_point2)</td>
<td>12</td>
</tr>
<tr>
<td>geom_pointrange2 (geom_point2)</td>
<td>12</td>
</tr>
<tr>
<td>geom_pointrange_borderless (geom_point2)</td>
<td>12</td>
</tr>
<tr>
<td>geom_pooljitter (geom_poolpoint)</td>
<td>13</td>
</tr>
<tr>
<td>geom_poolpoint</td>
<td>13</td>
</tr>
<tr>
<td>geom_violindot</td>
<td>15</td>
</tr>
<tr>
<td>geom_violinhalf</td>
<td>16</td>
</tr>
<tr>
<td>geoms_from_list (geom_from_list)</td>
<td>9</td>
</tr>
<tr>
<td>ggplot()</td>
<td>15, 17</td>
</tr>
<tr>
<td>ggplot2::geom_count()</td>
<td>13</td>
</tr>
<tr>
<td>ggplot2::geom_jitter()</td>
<td>13</td>
</tr>
<tr>
<td>ggplot2::geom_point()</td>
<td>13</td>
</tr>
<tr>
<td>ggplot2::geom_pointrange()</td>
<td>13</td>
</tr>
<tr>
<td>golden_ratio</td>
<td>18</td>
</tr>
<tr>
<td>layer()</td>
<td>16, 17</td>
</tr>
<tr>
<td>material_colors</td>
<td>19</td>
</tr>
<tr>
<td>metro_colors</td>
<td>19</td>
</tr>
<tr>
<td>modelbased::estimate_contrasts()</td>
<td>6</td>
</tr>
<tr>
<td>palette_bluebrown</td>
<td>20</td>
</tr>
<tr>
<td>palette_flat</td>
<td>20</td>
</tr>
<tr>
<td>palette_material</td>
<td>21</td>
</tr>
<tr>
<td>paletteMetro</td>
<td>21</td>
</tr>
<tr>
<td>palette_pizza</td>
<td>22</td>
</tr>
<tr>
<td>palette_see</td>
<td>22</td>
</tr>
<tr>
<td>palette_social</td>
<td>23</td>
</tr>
<tr>
<td>parameters::compare_parameters()</td>
<td>6</td>
</tr>
<tr>
<td>parameters::describe_distribution()</td>
<td>6</td>
</tr>
<tr>
<td>parameters::model_parameters()</td>
<td>6</td>
</tr>
<tr>
<td>parameters::n_clusters()</td>
<td>6</td>
</tr>
<tr>
<td>parameters::n_factors()</td>
<td>6</td>
</tr>
<tr>
<td>parameters::principal_components()</td>
<td>6</td>
</tr>
<tr>
<td>parameters::simulate_parameters()</td>
<td>6</td>
</tr>
<tr>
<td>patchwork::plot_layout()</td>
<td>53</td>
</tr>
<tr>
<td>performance::check_collinearity()</td>
<td>6</td>
</tr>
<tr>
<td>performance::check_heteroscedasticity()</td>
<td>6</td>
</tr>
<tr>
<td>performance::check_homogeneity()</td>
<td>6</td>
</tr>
<tr>
<td>performance::check_normality()</td>
<td>6</td>
</tr>
<tr>
<td>performance::check_outliers()</td>
<td>6</td>
</tr>
<tr>
<td>performance::check_posterior_predictions()</td>
<td>6</td>
</tr>
</tbody>
</table>

79
INDEX

80

performance::compare_performance(), 6
performance::performance_roc(), 6
pizza_colors, 23
plot.see_bayesfactor_models, 24
plot.see_bayesfactor_parameters, 25
plot.see_check_collinearity, 26
plot.see_check_distribution, 27
plot.see_check_heteroscedasticity, 27
plot.see_check_homogeneity, 28
plot.see_check_normality, 29
plot.see_check_outliers, 30
plot.see_compare_parameters, 31
plot.see_compare_performance, 32
plot.see_effectsize_table, 33
plot.see_equivalence_test
  (plot.see_equivalence_test_effectsize), 34
plot.see_equivalence_test_effectsize, 34
plot.see_equivalence_test_lm
  (plot.see_equivalence_test_effectsize), 34
plot.see_estimate_contrasts, 35
plot.see_estimate_density, 36
plot.see_hdi, 37
plot.see_n_factors, 38
plot.see_p_direction, 48
plot.see_p_significance, 49
plot.see_parameters_brms_meta, 39
plot.see_parameters_distribution, 41
plot.see_parameters_model, 42
plot.see_parameters_pca, 44
plot.see_parameters_sem
  (plot.see_parameters_model), 42
plot.see_parameters_simulate, 45
plot.see_performance_pp_check
  (print.see_performance_pp_check), 54
plot.see_performance_roc, 46
plot.see_point_estimate, 47
plot.see_rope, 50
plot.see_si, 52
plots, 53
print.see_performance_pp_check, 54

scale_color_bluebrown, 55
scale_color_bluebrown(), 20
scale_color_bluebrown_c
  (scale_color_bluebrown), 55
scale_color_bluebrown_d
  (scale_color_bluebrown), 55
scale_color_flat, 57
scale_color_flat(), 20
scale_color_flat_c (scale_color_flat), 57
scale_color_flat_d (scale_color_flat), 57
scale_color_material, 59
scale_color_material(), 21
scale_color_material_c
  (scale_color_material), 59
scale_color_material_d
  (scale_color_material), 59
scale_colorMetro, 61
scale_colorMetro(), 21
scale_colorMetro_c
  (scale_colorMetro), 61
scale_colorMetro_d
  (scale_colorMetro), 61
scale_color_pizza, 63
scale_color_pizza(), 22
scale_color_pizza_c
  (scale_color_pizza), 63
scale_color_pizza_d
  (scale_color_pizza), 63
scale_color_see, 65
scale_color_see(), 22
scale_color_see_c (scale_color_see), 65
scale_color_see_d (scale_color_see), 65
scale_color_social, 66
scale_color_social(), 23
scale_color_social_c
  (scale_color_social), 66
scale_color_social_d
  (scale_color_social), 66
scale_color_bluebrown
  (scale_color_bluebrown), 55
scale_color_bluebrown_c
  (scale_color_bluebrown), 55
scale_color_bluebrown_d
  (scale_color_bluebrown), 55
scale_colour_flat (scale_color_flat), 57
scale_colour_flat_c (scale_color_flat), 57
scale_colour_flat_d (scale_color_flat), 57
scale_colour_material