### Package ‘see’

April 9, 2021

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Title</strong></td>
<td>Visualisation Toolbox for 'easystats' and Extra Geoms, Themes and Color Palettes for 'ggplot2'</td>
</tr>
<tr>
<td><strong>Version</strong></td>
<td>0.6.3</td>
</tr>
<tr>
<td><strong>Maintainer</strong></td>
<td>Daniel Lüdecke <a href="mailto:d.luedecke@uke.de">d.luedecke@uke.de</a></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Provides plotting utilities supporting easystats-packages (<a href="https://github.com/easystats/easystats">https://github.com/easystats/easystats</a>) and some extra themes, geoms, and scales for 'ggplot2'. Color scales are based on <a href="https://www.materialui.co/colors">https://www.materialui.co/colors</a>.</td>
</tr>
<tr>
<td><strong>License</strong></td>
<td>GPL-3</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="https://easystats.github.io/see/">https://easystats.github.io/see/</a></td>
</tr>
<tr>
<td><strong>BugReports</strong></td>
<td><a href="https://github.com/easystats/see/issues">https://github.com/easystats/see/issues</a></td>
</tr>
<tr>
<td><strong>Depends</strong></td>
<td>graphics, grDevices, R (&gt;= 3.4), stats</td>
</tr>
<tr>
<td><strong>Imports</strong></td>
<td>bayestestR, dplyr, effectsize, ggplot2, ggridges, insight, magrittr, parameters (&gt;= 0.13.0), rlang</td>
</tr>
<tr>
<td><strong>Suggests</strong></td>
<td>brms, coda, correlation, emmeans, ggraph, ggrepel, glmMCMC, grid, gridExtra, htrr, lavaan, lme4, logspline, MASS, mclust, metafor, modelbased, NbClust, nFactors, performance (&gt;= 0.7.1), psych, qgplotr, randomForest, rmarkdown, rstanarm, splines, tidygraph, tidyr, spelling, testthat (&gt;= 3.0.0), vdiffr</td>
</tr>
<tr>
<td><strong>Encoding</strong></td>
<td>UTF-8</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>en-US</td>
</tr>
<tr>
<td><strong>RoxygenNote</strong></td>
<td>7.1.1</td>
</tr>
<tr>
<td><strong>Config/testthat/edition</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>NeedsCompilation</strong></td>
<td>no</td>
</tr>
<tr>
<td><strong>Author</strong></td>
<td>Daniel Lüdecke [aut, cre] (<a href="https://orcid.org/0000-0002-8895-3206">https://orcid.org/0000-0002-8895-3206</a>), Dominique Makowski [aut, inv] (<a href="https://orcid.org/0000-0001-5375-9967">https://orcid.org/0000-0001-5375-9967</a>), Mattan S. Ben-Shachar [aut] (<a href="https://orcid.org/0000-0002-4287-4801">https://orcid.org/0000-0002-4287-4801</a>), Indrajeet Patil [aut] (<a href="https://orcid.org/0000-0003-1995-6531">https://orcid.org/0000-0003-1995-6531</a>, @patilindrajeets),</td>
</tr>
</tbody>
</table>
R topics documented:

add_plot_attributes ........................................ 3
bluebrown_colors ........................................ 4
data_plot .................................................. 5
flat_colors .................................................. 8
geom_point2 ................................................ 8
geom_pooledpoint ......................................... 10
golden_ratio .............................................. 14
material_colors ........................................... 14
metro_colors ............................................... 15
palette_bluebrown ........................................ 16
data_plot .................................................. 5
palette_flat ............................................... 16
palette_material .......................................... 17
palette_metro .............................................. 17
data_plot .................................................. 5
palette_pizza ............................................. 18
palette_see ................................................ 18
palette_social ............................................ 19
pizza_colors ............................................... 19
plot.see_bayesfactor_models .............................. 20
plot.see_bayesfactor_parameters ......................... 21
plot.see_check_collinearity ................................ 22
plot.see_check_distribution .............................. 23
plot.see_check_heteroscedasticity ...................... 23
plot.see_check_homogeneity ................................ 24
plot.see_check_normality .................................. 25
plot.see_check_outliers .................................. 26
plot.see_cluster_analysis ................................ 27
plot.see_compare_parameters ............................. 28
plot.see_compare_performance ......................... 29
plot.see_easycormatrix ................................... 30
plot.see_easycorrelation ................................ 31
plot.see_effectsize ....................................... 31
plot.see_effectsize_table ................................ 31
plot.see_equivalence_test_effectsize .................. 32
plot.see_estimated_contrasts ............................ 33
plot.see_estimated_density .............................. 34
plot.see_hdi .............................................. 36
plot.see_n_factors ....................................... 37
add_plot_attributes

Description

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

Usage

```
add_plot_attributes(x)
```

Arguments

- `x` An object.
Examples

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

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

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

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

p + add_plot_attributes(data)

## End(Not run)

bluebrown_colors

---

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 color.

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

Examples

# Create a radar/spider chart with ggplot:
if (require("dplyr") && require("tidyr") && require("ggplot2")) {
  data <- iris %>%
    group_by(Species) %>%
    summarise_all(mean) %>%
    pivot_longer(~Species)

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

data_plot

Prepare objects for plotting or plot objects

Description

data_plot() attempts to extract and transform an object to be further plotted, while plot() tries to visualize results of functions from different packages of the easystats-project. See the documentation for your object's class:

- bayestestR::bayesfactor_models()
- bayestestR::bayesfactor_parameters()
- bayestestR::equivalence_test()
data_plot(x, data = NULL, ...)

Arguments

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

`data_plot()` is in most situation not needed when the purpose is plotting, since most `plot()`-functions in [see](https://example.com) 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,
    data = mtcars
  )

  x <- equivalence_test(model)
  plot(x)
}
```
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
flat_colors(...)

Arguments
...  Character names of colors.

Value
A character vector with color-codes.

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

geom_point2  Better looking points

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

Usage
geom_point2(..., stroke = 0, shape = 16)
geom_jitter2(..., size = 2, stroke = 0, shape = 16)
geom_pointrange2(..., stroke = 0)
geom_count2(..., stroke = 0)
Arguments

... Other arguments to be passed to `geom_point, geom_jitter, 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
geom_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
library(ggplot2)
library(see)

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

---

### geom_violindot

**Half-violin Half-dot plot**

#### Description

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

#### Usage

```r
geom_violindot(
  mapping = NULL,
  data = NULL,
  trim = TRUE,
  scale = "area",
  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 `aes()` or `aes()`. If specified and `inherit.aes` = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply `mapping` if there is no plot mapping.

- **data**: The data to be displayed in this layer. There are three options:
  - If NULL, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.
  - A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.
  - A function will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A function can be created from a formula (e.g. `~ head(.x, 10)`).
geom_violinhalf

trim
If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don’t trim the tails.

scale
if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.

show.legend
logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes
If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().

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.

size_dots, dots_size
Size adjustment for dots.

color_dots, dots_color
Color adjustment for dots.

fill_dots, dots_fill
Fill adjustment for dots.

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

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

Arguments

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

data The data to be displayed in this layer. There are three options:
  If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot().
  A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created.
  A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).

stat The statistical transformation to use on the data for this layer, as a string.

position Position adjustment, either as a string, or the result of a call to a position adjustment function.

trim If TRUE (default), trim the tails of the violins to the range of the data. If FALSE, don’t trim the tails.

scale if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.

show.legend logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.

inherit.aes If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn’t inherit behaviour from the default plot specification, e.g. borders().

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

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

Examples

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

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

---

### golden_ratio

**Golden Ratio**

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**

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

```r
material_colors(...)```
 metro_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(...)
palette_bluebrown  Blue-brown design color palette

Description
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

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

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

Arguments
- **palette**: Character name of palette. 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 (https://www.materialui.co/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 (https://www.materialui.co/metrocolors).

**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 on Social colors (https://www.materialui.co/socialcolors).

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

Arguments
```
...  
```
Character names of pizza ingredients.

Value
A character vector with color-codes.
plot.see_bayesfactor_models

Plot method for Bayes Factors for model comparison

Description

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` *Plotting model parameters* If NULL, coefficients are plotted in the order as they appear in the summary. Use `sort = "ascending"` (or `sort = TRUE`) resp. `sort = "descending"` to sort coefficients in ascending or descending order.
- `log` Show log-transformed Bayes factors.
- `prior_odds` optional vector of prior odds for the models. See BayesFactor::priorOdds. As the size of the pizza slices corresponds to posterior probability (which is a function of prior probability and the BF), custom `prior_odds` will change the slices’ size.
- `...` 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)
```

---

**plot.see_bayesfactor_parameters**

Plot method for Bayes Factors for a single parameter

**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` Size of point-geoms.
- `rope_color` Color of ROPE ribbon.
- `rope_alpha` Transparency level of ROPE ribbon.
- `show_intercept` Logical, if TRUE, the intercept-parameter is included in the plot. By default, it is hidden because in many cases the intercept-parameter has a posterior distribution on a very different location, so density curves of posterior distributions for other parameters are hardly visible.
- `...` Arguments passed to or from other methods.
Value

A ggplot2-object.

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"), ...)
```

Arguments

- `x`: An object.
- `data`: The original data used to create this object. Can be a statistical model or such.
- `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
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` : 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
library(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.

**Usage**

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

**Examples**

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

  x  
  data  

...  

Value

A ggplot2-object.

Examples

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

plot.see_check_homogeneity

*Plot method for homogeneity of variances checks*

Description

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

Usage

```r
## S3 method for class 'see_check_homogeneity'
plot(x, data = NULL, ...)
```

Arguments

  x  
  data  

...  

Value

A ggplot2-object.

Examples

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

Plot method for check model for (non-)normality of residuals

Description

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

Usage

## 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 or such.
size_line Size of line geoms.
size_point Size of point-geoms.
dot_alpha, alpha Alpha level of the confidence bands and point-geoms.
colors Character vector of length two, indicating the colors (in hex-format) for points and line.
detrend Should the plot be detrended?
... 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_normality(m)
plot(result)
```

Description

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

Usage

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

Arguments

- `x`: An object.
- `size_text`: Size of text labels.
- `size_line`: Size of line geoms.
- `dot_alpha`: ALpha level of the confidence bands and 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.
- `...`: Arguments passed to or from other methods.

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))
```

Description

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

Usage

```r
## S3 method for class 'see_cluster_analysis'
plot(x, data = NULL, n_columns = NULL, size_bar = 0.6, ...) 
```

Arguments

- `x` An object.
- `data` The original data used to create this object. Can be a statistical model or such.
- `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_bar` Size of bar geoms.
- `...` Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

```r
library(parameters)
groups <- cluster_analysis(iris[, 1:4], 3)
plot(groups)
```
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 = NULL,
  dodge_position = 0.8,
  sort = NULL,
  n_columns = NULL,
  ...)
```

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` Size of point-geoms.
- `size_text` Size of text labels.
- `dodge_position` Numeric, indicates the amount of "dodging" (spacing) between geoms.
- `sort` Plotting model parameters If NULL, coefficients are plotted in the order as they appear in the summary. Use `sort = "ascending"` (or `sort = TRUE`) resp. `sort = "descending"` to sort coefficients in ascending or descending order.
- `n_columns` Plotting Bayes factors Sort pie-slices by posterior probability (descending)? Numeric, indicates the amount of "dodging" (spacing) between geoms.
- `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("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)
}
```

Description

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

Usage

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

Arguments

- `x` An object.
- `size_line` Size of line geom.
- `...` 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_easycormatrix

Plot method for correlation matrices

Description

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

Usage

## S3 method for class 'see_easycormatrix'
plot(
  x,
  show_values = FALSE,
  show_p = FALSE,
  show_legend = TRUE,
  size_point = 1,
  size_text = 3.5,
  digits = 3,
  type = c("circle", "tile"),
  ...
)

Arguments

x An object.
show_values Logical, if TRUE, values are displayed.
show_p Logical, if TRUE, p-values or significant level is displayed.
show_legend Logical, show or hide legend.
size_point Size of point-geoms.
size_text Size of text labels.
digits Number of decimals used for values.
type Character vector, indicating the type of plot.
... Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

library(correlation)
data(mtcars)
result <- correlation(mtcars[, -c(8:9)])
s <- summary(result)
plot(s)
plot.see_easy.correlation

*Plot method for Gaussian Graphical Models*

**Description**

The plot() method for the `correlation::correlation()` function.

**Usage**

```r
## S3 method for class 'see_easy.correlation'
plot(x, size_point = 22, text_color = "white", node_color = "#647687", ...)
```

**Arguments**

- `x`: An object.
- `size_point`: Size of point-geoms.
- `text_color`: Color of text labels.
- `node_color`: Color of node- or circle-geoms.
- `...`: Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

```r
## Not run:
library(correlation)
library(ggraph)
result <- correlation(mtcars, partial = TRUE)
plot(result)
## End(Not run)
```

---

plot.see_effectsize_table

*Plot method for effect size tables*

**Description**

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

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, ...)
```

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

```r
## S3 method for class 'see_equivalence_test_lm'
plot(
  x,
  size_point = 0.7,
  ...)
Arguments

x An object.
... Arguments passed to or from other methods.
rope_color Color of ROPE ribbon.
rope_alpha Transparency level of ROPE ribbon.
show_intercept Logical, if TRUE, the intercept-parameter is included in the plot. By default, it is hidden because in many cases the intercept-parameter has a posterior distribution on a very different location, so density curves of posterior distributions for other parameters are hardly visible.
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 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, ...)

rope_color = "#0171D3",
rope_alpha = 0.2,
show_intercept = FALSE,
n_columns = 1,
... )
Arguments

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

Value

A ggplot2-object.

Examples

```r
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 `simulate_prior()`) and added to the plot.
- **priors_alpha** Alpha value of the prior distributions.
- **posteriors_alpha** Alpha value of the posterior distributions.
- **size_line** Size of line geoms.
- **size_point** Size of point-geoms.
- **centrality** The point-estimate (centrality index) to compute. May be "median", "mean" or "MAP".
- **ci** 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

```r
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

```r
## 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 or such.
- `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

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

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

```r
## 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 or such.
- `type` Character vector, indicating the type of plot.
- `size` Depending on type, 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,
  ...
)
```

Arguments

- `x` An object.
- `size_point` Size of point-geoms.
- `size_line` Size of line geoms.
- `size_text` Size of text labels.
- `posteriors_alpha` Alpha value of the posterior distributions.
- `rope_alpha` Transparency level of ROPE ribbon.
- `rope_color` 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`.
- `...` 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 `size_text = NULL` or `size_text = NA` 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)
}
```

```r
## End(Not run)
```

---

`plot.see_parameters_distribution`

*Plot method for describing distributions of vectors*
Description

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

Usage

```r
## 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`, will add range of dispersion for each variable to the plot.
- `dispersion_alpha`  
  Transparency level of dispersion ribbon.
- `dispersion_color`  
  Color of dispersion ribbon.
- `dispersion_style`  
  Character, style of dispersion area. "ribbon" for a ribbon, "curve" for a normal-curve.
- `size_bar`  
  Size of bar geoms.
- `highlight`  
  Vector with names of categories in `x` that should be highlighted.
- `highlight_color`  
  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.see_parameters_model

Plot method for model parameters

Description

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

Usage

## S3 method for class 'see_parameters_model'
plot(
x,  
show_intercept = FALSE,  
size_point = 0.8,  
size_text = NULL,  
sort = NULL,  
n_columns = NULL,  
type = c("forest", "funnel"),  
weight_points = TRUE,  
...  
)

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

size_text Size of text labels.
sort

**Plotting model parameters** If NULL, coefficients are plotted in the order as they appear in the summary. Use `sort = "ascending"` (or `sort = TRUE`) resp. `sort = "descending"` to sort coefficients in ascending or descending order.

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

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
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.

... Arguments passed to or from other methods.

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

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,
```
Arguments

x An object.
type Character vector, indicating the type of plot.
size_text Size of text labels.
text_color Color of text labels.
size Depending on type, size of bars, lines or segments.
... Arguments passed to or from other methods.

Value

A ggplot2-object.

Examples

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

```r
centrality = "median",
  ci = 0.95,
  ...
)
```

**Arguments**

- **x**  
  An object.

- **data**  
  The original data used to create this object. Can be a statistical model or such.

- **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**  
  Size of line geoms.

- **posteriors_alpha**  
  Alpha value of the posterior distributions.

- **centrality**  
  The point-estimate (centrality index) to compute. May be "median", "mean" or "MAP".

- **ci**  
  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**

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

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

**Usage**

```r
## 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
if (packageVersion("performance") > "0.7.0") {
  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)
}
```
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 or such.
- **size_point**: Size of point-geoms.
- **size_text**: 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 `simulate_prior()`) and added to the plot.
- **priors_alpha**: Alpha value of 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")
  result
  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 or such.
- `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 `simulate_prior()` and added to the plot.
- `priors_alpha`: Alpha value of the prior distributions.
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_glm(Sepal.Length ~ Petal.Width + Species, data = iris, refresh = 0)
  result <- p_direction(m)
  plot(result)
}
```

---

**Description**

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

**Usage**

```r
## S3 method for class 'see_p_significance'
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 or such.
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 `simulate_prior()` and added to the plot.

priors_alpha Alpha value of 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_glm(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

```r
## 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 or such.
- **rope_alpha**: Transparency level of ROPE ribbon.
- **rope_color**: 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

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

---

**plot.see_si**  
**Plot method for support intervals**

Description

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

Usage

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

Arguments

x An object.
si_color Color of SI ribbon.
si_alpha 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 Plot only the support data, or show the "raw" prior and posterior distributions? Only applies when plotting \texttt{si}.

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 <- si(m)
  result
  plot(result)
}
```

---

plots \hspace{1cm} \textit{Multiple plots side by side}

Description

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

Usage

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

Arguments

\begin{itemize}
  \item \texttt{...} grobs, gtables, ggplot or trellis objects
  \item \texttt{n_rows} Number of rows to align plots.
  \item \texttt{n_columns} Number of columns to align plots.
  \item \texttt{tags} Add tags to your subfigures. Can be \texttt{FALSE} (no tags), \texttt{TRUE} (letter tags) or character vector containing tags labels.
\end{itemize}
Examples

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

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

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

Description

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

Usage

```r
## S3 method for class 'see_performance_pp_check'
print(x, size_line = 0.7, line_alpha = 0.25, size_bar = 0.7, ...)

## S3 method for class 'see_performance_pp_check'
plot(x, size_line = 0.7, line_alpha = 0.25, size_bar = 0.7, ...)
```

Arguments

- `x` An object.
- `size_line` Size of line geoms.
- `line_alpha` Alpha value of lines indicating yrep.
- `size_bar` Size of bar geoms.
- `...` 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)
  pp_check(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,
  ...
)
```

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

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

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

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

```r
scale_colour_bluebrown_d(
  palette = "contrast",
  discrete = TRUE,
  reverse = FALSE,
  ...
)
```
\begin{verbatim}

    discrete = TRUE,
    reverse = FALSE,
    ...
  )

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

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

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()
\end{verbatim}
scale_color_flat

Flat UI color palette

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

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

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

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

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

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

scale_color_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).
scale_color_material

- **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 (https://www.materialui.co/color). 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,
  ...
)

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

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

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

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

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

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

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

- **palette** Character name of palette. 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

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

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

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

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

---

**scale_color_metro**

*Metro color palette*

Description

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

Usage

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

```r
scale_color_metro_d()
```
scale_color_metro

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

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

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

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

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

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

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

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

- **palette**: Character name of palette. 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

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

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

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

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

scale_color_pizza

Pizza color palette

Description

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

Usage

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

scale_color_pizza_d()
scale_color_pizza

    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)

# Example with discrete categories
ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_boxplot() +
  theme_modern() +
  scale_fill_pizza_d()

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

Description

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

Usage

```r
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,
```
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

```r
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 on Social (https://www.materialui.co/socialcolors). Use `scale_color_social_d` for *discrete* categories and `scale_color_social_c` for a *continuous* scale.
scale_color_social

Usage

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

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

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

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

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

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

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

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

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

**Arguments**

...  
Character names of colors.

**Value**
A character vector with color-codes.

**Examples**
```
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,
  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.
legend.position 
  the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)
theme_blackboard

Axis title spacing.

Legend elements text size in pts.

Legend elements text size in pts. Can be "none".

Axis title text size in pts.

Axis font face ("plain", "italic", "bold", "bold.italic").

Axis text size in pts.

Rotate the x axis labels.

Tags text size in pts.

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

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.
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**

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

**Arguments**

- **base_size** base font size, 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.
- **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".
theme_modern

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()

Description

A modern, sleek and elegant theme for ggplot.

Usage

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

base_size  base font size, 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.
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().
theme_radar

Usage

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

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

Arguments

base_size base font size, 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.
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

if (require("ggplot2") && require("dplyr") && require("tidyr")) {
  data <- iris %>%
    group_by(Species) %>%
    summarise_all(mean) %>%
    pivot_longer(-Species)

  data %>%
    ggplot(aes(
      x = name,
      y = value,
      color = Species,
      group = Species,
      fill = Species
    )) +
    geom_polygon(size = 1, alpha = .1) +
    coord_radar() +
    theme_radar()
}
add_plot_attributes, 3
aes(), 11, 13
aes(), 11, 13
bayestestR::bayesfactor_models(), 5
bayestestR::bayesfactor_parameters(), 5
bayestestR::equivalence_test(), 5
bayestestR::estimate_density(), 6
bayestestR::hdi(), 6
bayestestR::p_direction(), 6
bayestestR::p_significance(), 6
bayestestR::si(), 6
bluebrown_colors, 4
borders(), 12, 13
colorRampPalette(), 16–19, 54, 58, 60, 62, 63, 65
coord_radar, 5, 74
correlation::correlation(), 6
data_plot, 3, 5
effectsize::effectsize(), 6
flat_colors, 8
fortify(), 11, 13
geom_count2(geom_point2), 8
geom_count_borderless(geom_point2), 8
geom_jitter(), 9
geom_jitter2(geom_point2), 8
geom_jitter_borderless(geom_point2), 8
geom_point(), 9
geom_point2, 8
geom_point_borderless(geom_point2), 8
geom_pointrange, 9
geom_pointrange2(geom_point2), 8
geom_pointrange_borderless (geom_point2), 8
geom_pooljitter(geom_poolpoint), 10
geom_poolpoint, 10
geom_violindot, 11
geom_violinhalf, 12
ggplot(), 11, 13
ggplot2::geom_count, 9
golden_ratio, 14
layer(), 12, 13
material_colors, 14
metro_colors, 15
modelbased::estimate_contrasts(), 6
palette_bluebrown, 16
palette_flat, 16
palette_material, 17
palette_metro, 17
palette_pizza, 18
palette_see, 18
palette_social, 19
parameters::cluster_analysis(), 6
parameters::compare_parameters(), 6
parameters::describe_distribution(), 6
parameters::model_parameters(), 6
parameters::n_clusters(), 6
parameters::n_factors(), 6
parameters::principal_components(), 6
parameters::simulate_parameters(), 6
performance::check_collinearity(), 6
performance::check_heteroscedasticity(), 6
performance::check_homogeneity(), 6
performance::check_normality(), 6
performance::check_outliers(), 6
performance::compare_performance(), 6
performance::performance_roc(), 6
performance::pp_check(), 6
pizza_colors, 19
plot.see_bayesfactor_models, 20
plot.see_bayesfactor_parameters, 21
INDEX

plot.see_check_collinearity, 22
plot.see_check_distribution, 23
plot.see_check_heteroscedasticity, 23
plot.see_check_homogeneity, 24
plot.see_check_normality, 25
plot.see_check_outliers, 26
plot.see_cluster_analysis, 27
plot.see_compare_parameters, 28
plot.see_compare_performance, 29
plot.see_easy_correlation, 30
plot.see_easycorrelation, 31
plot.see_effectsize_table, 31
plot.see_equivalence_test
   (plot.see_equivalence_test_effectsize)
   plot.see_equivalence_test_effectsize,
   32
plot.see_equivalence_test_lmm
   (plot.see_equivalence_test_effectsize)
   plot.see_equivalence_test_effectsize,
   32
plot.see_estimate_contrasts, 33
plot.see_estimate_density, 34
plot.see_hdidi, 36
plot.see_n_factors, 37
plot.see_p_direction, 47
plot.see_p_significance, 48
plot.see_parameters_brms_meta, 38
plot.see_parameters_distribution, 39
plot.see_parameters_model, 41
plot.see_parameters_pca, 42
plot.see_parameters_sem
   (plot.see_parameters_model), 41
plot.see_parameters_simulate, 43
plot.see_performance_pp_check
   (print.see_performance_pp_check),
   52
plot.see_performance Roc, 45
plot.see_point_estimate, 46
plot.see_rope, 49
plot.see_si, 50
plots, 51
print.see_performance_pp_check, 52

scale_color_bluebrown, 53
scale_color_bluebrown(), 16
scale_color_bluebrown_c
   (scale_color_bluebrown), 53
scale_color_bluebrown_d
   (scale_color_bluebrown), 53
scale_color_flat, 55
scale_color_flat()
   scale_color_flat_c(scale_color_flat),
   55
scale_color_flat_d(scale_color_flat),
   55
scale_color_material, 56
scale_color_material()
   scale_color_material_c
   (scale_color_material), 56
scale_color_material_d
   (scale_color_material), 56
scale_color_material()  (scale_color_material), 17
scale_color_material_c
   (scale_color_material), 58
scale_color_material_d
   (scale_color_material), 58
scale_color_material
   scale_color_material(), 18
scale_color_material_c
   (scale_color_material), 60
scale_color_material_d
   (scale_color_material), 60
scale_color_material
   scale_color_material(), 60
scale_color_material_c
   (scale_color_material), 63
scale_color_material_d
   (scale_color_material), 63
scale_color_material
   scale_color_material(), 63
scale_colour_bluebrown
   (scale_color_bluebrown), 53
scale_colour_bluebrown_c
   (scale_color_bluebrown), 53
scale_colour_bluebrown_d
   (scale_color_bluebrown), 53
scale_colour_flat
   (scale_color_flat), 55
scale_colour_flat_c
   (scale_color_flat), 55
scale_colour_flat_d
   (scale_color_flat), 55
scale_colour_material
   (scale_color_material), 56
scale_colour_material_c
(scale_color_material), 56
scale_colour_material_d
(scale_color_material), 56
scale_colour_metro (scale_color_metro), 58
scale_colour_metro_c
(scale_color_metro), 58
scale_colour_metro_d
(scale_color_metro), 58
scale_colour_pizza (scale_color_pizza), 60
scale_colour_pizza_c
(scale_color_pizza), 60
scale_colour_pizza_d
(scale_color_pizza), 60
scale_colour_see (scale_color_see), 62
scale_colour_see_c (scale_color_see), 62
scale_colour_see_d (scale_color_see), 62
scale_fill_social (scale_color_social), 63
scale_fill_social_c
(scale_color_social), 63
scale_fill_social_d
(scale_color_social), 63
scale_fill_see (scale_color_see), 62
scale_fill_see_c (scale_color_see), 62
scale_fill_see_d (scale_color_see), 62
scale_fill_material (scale_color_material), 56
scale_fill_material_c
(scale_color_material), 56
scale_fill_material_d
(scale_color_material), 56
scale_fill_metro (scale_color_metro), 58
scale_fill_metro_c (scale_color_metro), 58
scale_fill_metro_d (scale_color_metro), 58
scale_fill_pizza (scale_color_pizza), 60
scale_fill_pizza_c (scale_color_pizza), 60
scale_fill_pizza_d (scale_color_pizza), 60
see_colors, 66
si, 51
simulate_prior(), 35, 46, 47, 49
social_colors, 66
theme_abyss, 67
theme_blackboard, 68
theme_lucid, 70
theme_modern, 71
theme_radar, 72
theme_radar_dark (theme_radar), 72