Package ‘see’

July 17, 2024

Type Package

Title Model Visualisation Toolbox for 'easystats' and 'ggplot2'

Version 0.8.5

Maintainer Indrajeet Patil <patilindrajeet.science@gmail.com>

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


License MIT + file LICENSE

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

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

Depends graphics, grDevices, R (>= 4.0), stats

Imports bayestestR (>= 0.13.2), correlation (>= 0.8.5), datawizard (>= 0.12.1), effectsize (>= 0.8.9), ggplot2, insight (>= 0.20.2), modelbased (>= 0.8.8), parameters (>= 0.22.0), performance (>= 0.12.1)

Suggests brms, curl, DHARMa, emmeans, factoextra, ggdist, ggraph, ggrepel, ggridges, ggside, glmmTMB, grid, htr, htr2, lavaan, lme4, logspline, MASS, mclust, merDeriv, mgcv, metafor, NbClust, nFactors, patchwork (>= 1.2.0), poorman, psych, qqplotr (>= 0.0.6), randomForest, RcppEigen, rlang, rmarkdown, rstanarm, scales (>= 1.3.0), splines, testthat (>= 3.2.1), tidygraph, vdiffr (>= 1.0.7)

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/testthat/parallel true

Config/Needs/website easystats/easystatstemplate
Config/rcmdcheck/ignore-inconsequential-notes  true

NeedsCompilation  no

Author  Daniel Lüdecke [aut, ctb] (<https://orcid.org/0000-0002-8895-3206>, @strengejacke),
Dominique Makowski [aut, inv] (<https://orcid.org/0000-0001-5375-9967>, @Dom_Makowski),
Indrajeet Patil [aut, cre] (<https://orcid.org/0000-0003-1995-6531>, @patilindrajeets),
Mattan S. Ben-Shachar [aut, ctb]
(<https://orcid.org/0000-0002-4287-4801>, @mattansb),
Brenton M. Wiernik [aut, ctb] (<https://orcid.org/0000-0001-9560-6336>),
Philip Waggoner [aut, ctb] (<https://orcid.org/0000-0002-7825-7573>),
Jeffrey R. Stevens [ctb] (<https://orcid.org/0000-0003-2375-1360>),
Matthew Smith [rev] (@SmithMatt90),
Jakob Bossek [rev] (@BossekJakob)

Repository  CRAN

Date/Publication  2024-07-17 20:30:01 UTC

Contents

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add_plot_attributes</td>
<td>4</td>
</tr>
<tr>
<td>bluebrown_colors</td>
<td>5</td>
</tr>
<tr>
<td>coord_radar</td>
<td>5</td>
</tr>
<tr>
<td>data_plot</td>
<td>6</td>
</tr>
<tr>
<td>flat_colors</td>
<td>8</td>
</tr>
<tr>
<td>geom_binomdensity</td>
<td>9</td>
</tr>
<tr>
<td>geom_from_list</td>
<td>10</td>
</tr>
<tr>
<td>geom_point2</td>
<td>12</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>17</td>
</tr>
<tr>
<td>golden_ratio</td>
<td>19</td>
</tr>
<tr>
<td>material_colors</td>
<td>20</td>
</tr>
<tr>
<td>metro_colors</td>
<td>20</td>
</tr>
<tr>
<td>okabeito_colors</td>
<td>21</td>
</tr>
<tr>
<td>palette_bluebrown</td>
<td>22</td>
</tr>
<tr>
<td>palette_colorhex</td>
<td>22</td>
</tr>
<tr>
<td>palette_flat</td>
<td>23</td>
</tr>
<tr>
<td>palette_material</td>
<td>23</td>
</tr>
<tr>
<td>palette_metro</td>
<td>24</td>
</tr>
<tr>
<td>palette_okabeito</td>
<td>24</td>
</tr>
<tr>
<td>palette_pizza</td>
<td>25</td>
</tr>
<tr>
<td>palette_see</td>
<td>26</td>
</tr>
<tr>
<td>palette_social</td>
<td>26</td>
</tr>
<tr>
<td>pizza_colors</td>
<td>27</td>
</tr>
<tr>
<td>plot.datawizard_tables</td>
<td>27</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>plot.see_bayesfactor_models</td>
<td>28</td>
</tr>
<tr>
<td>plot.see_bayesfactor_parameters</td>
<td>30</td>
</tr>
<tr>
<td>plot.see_check_collinearity</td>
<td>31</td>
</tr>
<tr>
<td>plot.see_check_distribution</td>
<td>32</td>
</tr>
<tr>
<td>plot.see_check_heteroscedasticity</td>
<td>32</td>
</tr>
<tr>
<td>plot.see_check_homogeneity</td>
<td>33</td>
</tr>
<tr>
<td>plot.see_check_model</td>
<td>34</td>
</tr>
<tr>
<td>plot.see_check_normality</td>
<td>35</td>
</tr>
<tr>
<td>plot.see_check_outliers</td>
<td>37</td>
</tr>
<tr>
<td>plot.see_compare_parameters</td>
<td>38</td>
</tr>
<tr>
<td>plot.see_compare_performance</td>
<td>39</td>
</tr>
<tr>
<td>plot.see_effectsize_table</td>
<td>40</td>
</tr>
<tr>
<td>plot.see_equivalence_test_effectsize</td>
<td>41</td>
</tr>
<tr>
<td>plot.see_estimate_contrasts</td>
<td>42</td>
</tr>
<tr>
<td>plot.see_estimate_density</td>
<td>43</td>
</tr>
<tr>
<td>plot.see_hdi</td>
<td>44</td>
</tr>
<tr>
<td>plot.see_n_factors</td>
<td>45</td>
</tr>
<tr>
<td>plot.see_parameters_brms_meta</td>
<td>46</td>
</tr>
<tr>
<td>plot.see_parameters_distribution</td>
<td>48</td>
</tr>
<tr>
<td>plot.see_parameters_model</td>
<td>49</td>
</tr>
<tr>
<td>plot.see_parameters_pca</td>
<td>51</td>
</tr>
<tr>
<td>plot.see_parameters_simulate</td>
<td>52</td>
</tr>
<tr>
<td>plot.see_performance_roc</td>
<td>54</td>
</tr>
<tr>
<td>plot.see_performance_simres</td>
<td>54</td>
</tr>
<tr>
<td>plot.see_point_estimate</td>
<td>56</td>
</tr>
<tr>
<td>plot.see_p_direction</td>
<td>57</td>
</tr>
<tr>
<td>plot.see_p_function</td>
<td>58</td>
</tr>
<tr>
<td>plot.see_p_significance</td>
<td>60</td>
</tr>
<tr>
<td>plot.see_rope</td>
<td>61</td>
</tr>
<tr>
<td>plot.see_si</td>
<td>62</td>
</tr>
<tr>
<td>plots</td>
<td>63</td>
</tr>
<tr>
<td>print.see_performance_pp_check</td>
<td>64</td>
</tr>
<tr>
<td>scale_color_bluebrown</td>
<td>66</td>
</tr>
<tr>
<td>scale_color_colorhex</td>
<td>68</td>
</tr>
<tr>
<td>scale_color_flat</td>
<td>71</td>
</tr>
<tr>
<td>scale_color_material</td>
<td>73</td>
</tr>
<tr>
<td>scale_color.metro</td>
<td>76</td>
</tr>
<tr>
<td>scale_color_okabeito</td>
<td>78</td>
</tr>
<tr>
<td>scale_color_pizza</td>
<td>80</td>
</tr>
<tr>
<td>scale_color_see</td>
<td>83</td>
</tr>
<tr>
<td>scale_color_social</td>
<td>85</td>
</tr>
<tr>
<td>see_colors</td>
<td>87</td>
</tr>
<tr>
<td>social_colors</td>
<td>88</td>
</tr>
<tr>
<td>theme_abyss</td>
<td>89</td>
</tr>
<tr>
<td>theme_blackboard</td>
<td>90</td>
</tr>
<tr>
<td>theme_lucid</td>
<td>92</td>
</tr>
<tr>
<td>theme_modern</td>
<td>93</td>
</tr>
<tr>
<td>theme_radar</td>
<td>95</td>
</tr>
</tbody>
</table>
add_plot_attributes

Description

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

`add_plot_attributes(x)`

Arguments

- `x` An object.

Examples

```r
library(rstanarm)
library(bayestestR)
library(see)
library(ggplot2)

model <- suppressWarnings(stan_glm(
  Sepal.Length ~ Petal.Width + Species + Sepal.Width,
  data = iris,
  chains = 2, iter = 200, refresh = 0
))

result <- bayestestR::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
p + add_plot_attributes(data)
```
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

library(ggplot2)

# Create a radar/spider chart with ggplot:
data(iris)
data <- aggregate(iris[-5], list(Species = iris$Species), mean)
data <- datwizard::data_to_long(
data,
)
ggplot(data, aes(x = name, y = value, color = Species, group = Species)) +
  geom_polygon(fill = NA, linewidth = 2) +
  coord_radar(start = -pi / 4)

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_predictions()

Usage

data_plot(x, ...)

## S3 method for class 'compare_performance'
data_plot(x, data = NULL, ...)

Arguments

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

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

library(bayestestR)
library(rstanarm)

model <<- suppressWarnings(stan_glm(
  Sepal.Length ~ Petal.Width * Species,
  data = iris,
  chains = 2, iter = 200, refresh = 0
))
```r
x <- rope(model, verbose = FALSE)
plot(x)

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

x <- p_direction(model, verbose = FALSE)
plot(x)

model <<- suppressWarnings(stan_glm(
  mpg ~ wt + gear + cyl + disp,
  chains = 2,
  iter = 200,
  refresh = 0,
  data = mtcars
))

x <- equivalence_test(model, verbose = FALSE)
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**

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

**Arguments**

```r
... Character names of colors.
```

**Value**

A character vector with color-codes.

**Examples**

```r
flat_colors()

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

Description

Add dot-densities for binary y variables

Usage

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 = "proportion")
```
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) +
geom_from_list(l2)
ggplot() +
  geoms_from_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
)
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")
  ))

 ggplot() +
  geom_from_list(list(
    geom = "density_2d_raster", 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_from_list(list(geom = "scale_color_material_d", palette = "rainbow")) +
  geom_from_list(list(geom = "theme_void"))

# Example 5 (Smooths and side densities) --------------------------

ggplot(iris, aes(x = Sepal.Length, y = Petal.Width)) +
  geom_from_list(list(geom = "point")) +
  geom_from_list(list(aes = list(x = "Sepal.Length"), geom = "ggside::geom_xsidedensity")) +
  geom_from_list(list(aes = list(x = "Sepal.Length"), geom = "ggside::scale_xsidey_continuous", breaks = NULL))

---

**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)

geom_count_borderless(..., stroke = 0)

geom_point_borderless(...)
geom_poolpoint

geom_jitter_borderless(...)  
geom_pointrange_borderless(...)

Arguments

... Other arguments to be passed to \texttt{ggplot2::geom_point()}, \texttt{ggplot2::geom_jitter()}, \texttt{ggplot2::geom_pointrange()}, or \texttt{ggplot2::geom_count()}.

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

Note

The color aesthetics for \texttt{geom_point_borderless()} is “fill”, not "color". See 'Examples'.

Examples

library(ggplot2)
library(see)

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

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

plots(normal, new, n_columns = 2)

 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  \hspace{1cm} Pool ball points

Description

Points labelled with the observation name.
Usage

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

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

Arguments

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

Examples

```
library(ggplot2)
library(see)

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

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

Half-violin Half-dot plot

Description

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

Usage

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

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

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

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

show.legend

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

inherit.aes

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

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()’s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can not be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

• Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom’s documentation has an Aesthetics section that lists the available options. The ’required’ aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.

• When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is stat_density(geom = "area", outline.type = "both"). The geom’s documentation lists which parameters it can accept.

• Inversely, when constructing a layer using a geom_*() function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is geom_area(stat = "density", adjust = 0.5). The stat’s documentation lists which parameters it can accept.

• The key_glyph argument of layer() may also be passed on through .... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

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)
```r
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**  
```r
geom_violinhalf(
  mapping = NULL,
  data = NULL,
  stat = "ydensity",
  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 `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. When using a `geom_*()` function to construct a layer, the `stat` argument can be used to override the default coupling between geoms and stats. The `stat` argument accepts the following:
• A Stat ggproto subclass, for example StatCount.
• A string naming the stat. To give the stat as a string, strip the function name of the stat_prefix. For example, to use stat_count(), give the stat as "count".
• For more information and other ways to specify the stat, see the layer stat documentation.

**position**
A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The position argument accepts the following:
• The result of calling a position function, such as position_jitter(). This method allows for passing extra arguments to the position.
• A string naming the position adjustment. To give the position as a string, strip the function name of the position_prefix. For example, to use position_jitter(), give the position as "jitter".
• For more information and other ways to specify the position, see the layer position documentation.

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

**flip**
Should the half-violin plot switch directions? By default, this is FALSE and all half-violin geoms will have the flat-side on facing leftward. If 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? 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()’s params argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the position argument, or aesthetics that are required can not be passed through .... Unknown arguments that are not part of the 4 categories below are ignored.

• Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, colour = "red" or linewidth = 3. The geom’s documentation has an Aesthetics section that lists the available options. The 'required' aesthetics cannot be passed on to the params. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
• When constructing a layer using a stat_*() function, the ... argument can be used to pass on parameters to the geom part of the layer. An example
of this is `stat_density(geom = "area", outline.type = "both")`. The geom’s documentation lists which parameters it can accept.

- Inversely, when constructing a layer using a `geom_*()` function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat’s documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through ... This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

**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`:
```r
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
```r
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**

<table>
<thead>
<tr>
<th>golden_ratio</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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 colors.

**Usage**

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

**Arguments**

```r
...
```
Character names of colors.

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

**Examples**

```r
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 colors.

**Usage**

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

**Arguments**

```r
...
```
Character names of colors.
okabeito_colors

Value
A character vector with color-codes.

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

okabeito_colors Extract Okabe-Ito colors as hex codes

Description
Can be used to get the hex code of specific colors from the Okabe-Ito palette. Use okabeito_colors() to see all available colors.

Usage
okabeito_colors(..., original_names = FALSE, black_first = FALSE, amber = TRUE)
oi_colors(..., original_names = FALSE, black_first = FALSE, amber = TRUE)

Arguments
... Character names of colors.
original_names Logical. Should the colors be named using the original names used by Okabe and Ito (2008), such as "vermillion" (TRUE), or simplified names, such as "red" (FALSE, default)? Only used if no colors are specified (to see all available colors).
black_first Logical. Should black be first (TRUE) or last (FALSE, default) in the color palette? Only used if no colors are specified (to see all available colors).
amber If amber color should replace yellow in the palette.

Value
A character vector with color-codes.

Examples
okabeito_colors()
okabeito_colors(c("red", "light blue", "orange"))
okabeito_colors(original_names = TRUE)
okabeito_colors(black_first = TRUE)
palette_bluebrown  

*Blue-brown design color palette*

**Description**

The palette based on blue-brown colors.

**Usage**

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

**Arguments**

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

*Color palettes from https://www.color-hex.com/*

**Description**

This function downloads a requested color palette from [https://www.color-hex.com/](https://www.color-hex.com/). This website provides a large number of user-submitted color palettes.

**Usage**

```r
palette_colorhex(palette = 1014416, reverse = FALSE, ...)
```

**Arguments**

- **palette**  
  The numeric code for a palette at [https://www.color-hex.com/](https://www.color-hex.com/). For example, 1014416 for the Josiah color palette (number 1014416).
- **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_colorhex()`.
palette_flat

Note
The default Josiah color palette (number 1014416) is available without an internet connection. All other color palettes require an internet connection to download and access.

---

palette_flat  Flat UI color palette

Description
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", "light" (for dark themes), "black_first", full_original, or black_first_original.
- **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
palette_material(palette = "contrast", reverse = FALSE, ...)

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

*Okabe-Ito color palette*

Description

The palette based proposed by Okabe and Ito (2008).

Usage

```r
palette_okabeito(palette = "full_amber", reverse = FALSE, order = 1:9, ...)
```

```r
palette_oi(palette = "full_amber", reverse = FALSE, order = 1:9, ...)
```

"full_amber", reverse = FALSE, order = 1:9, ...)
**palette_pizza**

### Arguments

- **palette**: Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast", "light" (for dark themes), "black_first", full_original, or black_first_original.
- **reverse**: Boolean indicating whether the palette should be reversed.
- **order**: A vector of numbers from 1 to 9 indicating the order of colors to use (default: 1:9)
- **...**: Additional arguments to pass to `colorRampPalette()`.

### Details

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

### References


---

**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", "light" (for dark themes), "black_first", `full_original`, or `black_first_original`.
- `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", "light" (for dark themes), "black_first", `full_original`, or `black_first_original`.
- `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**

```r
...
Character names of pizza ingredients.
```

**Value**

A character vector with color-codes.

---

**plot.datawizard_tables**

Plot tabulated data.

---

**Description**

Plot tabulated data.

**Usage**

```r
## S3 method for class 'datawizard_tables'
plot(
  x,
  label_values = TRUE,
  show_na = c("if_any", "always", "never"),
  na_label = "(Missing)",
  error_bar = TRUE,
  ci = 0.95,
  fill_col = "#87CEFA",
  color_error_bar = "#607B8B",
  ...
)
```

```r
## S3 method for class 'datawizard_table'
plot(
  x,
  label_values = TRUE,
  ```
show_na = c("if_any", "always", "never"),
na_label = "(Missing)",
error_bar = TRUE,
ci = 0.95,
fill_col = "#87CEFA",
color_error_bar = "#607B8B",
...)

Arguments

x Object created by datawizard::data_tabulate().

label_values Logical. Should values and percentages be displayed at the top of each bar.

show_na Should missing values be dropped? Can be "if_any" (default) to show the missing category only if any missing values are present, "always" to always show the missing category, or "never" to never show the missing category.

na_label The label given to missing values when they are shown.

error_bar Logical. Should error bars be displayed? If TRUE, confidence intervals computed using the Wilson method are shown. See Brown et al. (2001) for details.

ci Confidence Interval (CI) level. Defaults to 0.95 (95%).

fill_col Color to use for category columns (default: "#87CEFA").

color_error_bar Color to use for error bars (default: "#607B8B").

... Unused

References


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

## 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, respec-
     tively. Setting sort = TRUE is the same as sort = "ascending".
   • Plotting Bayes factors: Sort pie-slices by posterior probability (descend-
     ing)?

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

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**: 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.

Value

A ggplot2-object.
plot.see_check_collinearity

---

**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 = 3.5,
  size_line = 0.8,
  size_title = 12,
  size_axis_title = base_size,
  base_size = 10,
  ...
)
```

**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 specifying size of line geoms.
- `base_size`, `size_axis_title`, `size_title` Numeric value specifying size of axis and plot titles.
- `...` 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
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,
  size_point = 2,
  size_line = 0.8,
  size_title = 12,
  size_axis_title = base_size,
  base_size = 10,
  ...
)
```

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_line`  
  Numeric value specifying size of line geoms.
- `base_size`, `size_axis_title`, `size_title`  
  Numeric value specifying size of axis and plot titles.
- `...`  
  Arguments passed to or from other methods.

Value

A ggplot2-object.

See Also

See also the vignette about `check_model()`.

Examples

```r
m <- lm(mpg ~ wt + cyl + gear + disp, data = mtcars)
result <- performance::check_heteroscedasticity(m)
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.
## plot.see_check_homogeneity

### Usage

```
## S3 method for class 'see_check_homogeneity'
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

```r
library(performance)

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

---

**plot.see_check_model**  
Plot method for checking model assumptions

### Description

The `plot()` method for the `performance::check_model()` function. Diagnostic plots for regression models.

### Usage

```
## S3 method for class 'see_check_model'
plot(
  x,
  style = theme_lucid,
  colors = NULL,
  type = c("density", "discrete_dots", "discrete_interval", "discrete_both"),
  n_columns = 2,
  ...)
```
plot.see_check_normality

Arguments

- **x**: An object.
- **style**: A ggplot2-theme.
- **colors**: Character vector of length two, indicating the colors (in hex-format) for points and line.
- **type**: Plot type for the posterior predictive checks plot. Can be “density” (default), “discrete_dots”, “discrete_interval” or “discrete_both” (the discrete_* options are appropriate for models with discrete - binary, integer or ordinal etc. - outcomes).
- **n_columns**: Number of columns to align plots.
- **...**: Arguments passed to or from other methods.

Value

A ggplot2-object.

See Also

See also the vignette about `check_model()`.

Examples

```r
library(performance)

model <- lm(qsec ~ drat + wt, data = mtcars)
plot(check_model(model))
```

---

**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

```r
# S3 method for class 'see_check_normality'
plot(
  x,
  type = c("qq", "pp", "density"),
  data = NULL,
  size_line = 0.8,
  size_point = 2,
  size_title = 12,
```
Arguments

- **x**: An object.
- **type**: Character vector, indicating the type of plot. Options are "qq" (default) for quantile-quantile (Q-Q) plots, "pp" for probability-probability (P-P) plots, or "density" for density overlay plots.
- **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.
- **base_size**, **size_axis_title**, **size_title**: Numeric value specifying size of axis and plot titles.
- **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 Q-Q and P-P plots should be de-trended (also known as worm plots).
- **method**: The method used for estimating the qq/pp bands. Default to "ell" (equal local levels / simultaneous testing - recommended). Can also be one of "pointwise" or "boot" for pointwise confidence bands, or "ks" or "ts" for simultaneous testing. See `ggplotr::stat_qq_band()` for details.
- **...**: Arguments passed to or from other methods.

Value

A `ggplot2`-object.

See Also

See also the vignette about `check_model()`.

Examples

```r
library(performance)

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

plot(result, type = "qq", detrend = TRUE)

---

plot.see_check_outliers

Plot method for checking outliers

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,
  size_title = 12,
  size_axis_title = base_size,
  base_size = 10,
  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.
- `base_size, size_axis_title, size_title` Numeric value specifying size of axis and plot titles.
- `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.
**plot.see_compare_parameters**

::

     type  Character vector, indicating the type of plot. Options are "dots" (default) for a scatterplot of leverage (hat) values versus residuals, with Cook's Distance contours for evaluating influential points, or "bars" for a bar chart of (rescaled) outlier statistic values for each data point. Only used for outlier plots of fitted models; for outlier plots of raw data values, type = "bars" is always used.

     show_labels Logical. If TRUE, text labels are displayed.

... Arguments passed to or from other methods.

### Value

A ggplot2-object.

### Examples

```
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

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 <- parameters::compare_parameters(lm1, lm2, lm3)
plot(result)
Usage

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

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

```r
library(modelbased)

model <- lm(Sepal.Width ~ Species, data = iris)
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.
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.

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

ci Numeric value of probability of the CI (between 0 and 1) to be estimated. Default to 0.95.

... Arguments passed to or from other methods.

Value
A ggplot2-object.

Examples

```r
library(rstanarm)
library(bayestestR)
set.seed(123)
m <<- suppressWarnings(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::hdif()` and related function.

**Usage**

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

library(rstanarm)
library(bayestestR)
set.seed(123)
m <- suppressWarnings(stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0))
result <- bayestestR::hdi(m)
result
plot(result)
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. Options are three different shapes to illustrate the degree of consensus between dimensionality methods for each number of factors: "bar" (default) for a bar chart, "line" for a horizontal point and line chart, or "area" for an area chart (frequency polygon).
- **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
data(mtcars)
result <- parameters::n_factors(mtcars, type = "PCA")
result

plot(result) # type = "bar" by default
plot(result, type = "line")
plot(result, type = "area")
```

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.
posteriorAlpha 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

library(parameters)
library(brms)
library(metafor)
data(dat.bcg)

dat <- escalc(
  measure = "RR",
  ai = tpos,
  bi = tneg,


```r

# see_parameters_distribution

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, a range of dispersion for each variable to the plot will be added.

```r

# model
set.seed(123)
priors <- c(
  prior(normal(0, 1), class = Intercept),
  prior(cauchy(0, 0.5), class = sd)
)
model <- suppressWarnings(
  brm(yi | se(vi) ~ 1 + (1 | author), data = dat, refresh = 0, silent = 2)
)

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

```r

```
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

library(parameters)
set.seed(333)
x <- sample(1:100, 1000, replace = TRUE)
result <- describe_distribution(x)
result
plot(result)
weight_points = TRUE,
show_labels = FALSE,
show_estimate = TRUE,
show_interval = TRUE,
show_density = FALSE,
log_scale = FALSE,
...)

## 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
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
## 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. Options are three different shapes to represent component loadings: "bar" (default) for a horizontal bar chart, or "line" for a horizontal point and line chart.
- **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)
```

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".
- **ci**: Numeric value of probability of the CI (between 0 and 1) to be estimated. Default to 0.95.
- **...**: 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)
```
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

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_performance_simres

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

Description

The plot() method for the performance::check_residuals() resp. performance::simulate_residuals() function.
Usage

```r
## S3 method for class 'see_performance_simres'
plot(
  x,
  size_line = 0.8,
  size_point = 2,
  size_title = 12,
  size_axis_title = base_size,
  base_size = 10,
  alpha = 0.2,
  dot_alpha = 0.8,
  colors = c("#3aaf85", "#1b6ca8"),
  detrend = FALSE,
  transform = NULL,
  style = theme_lucid,
  ...
)
```

Arguments

- `x`: An object.
- `size_line`: Numeric value specifying size of line geoms.
- `size_point`: Numeric specifying size of point-geoms.
- `base_size`, `size_axis_title`, `size_title`: Numeric value specifying size of axis and plot titles.
- `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 Q-Q and P-P plots should be de-trended (also known as worm plots).
- `transform`: Function to transform the residuals. If `NULL` (default), no transformation is applied and uniformly distributed residuals are expected. See argument quantileFuntion in `?DHARMa:::residuals.DHARMa` for more details.
- `style`: A ggplot2-theme.
- `...`: Arguments passed to or from other methods.

Value

A ggplot2-object.

See Also

See also the vignette about `check_model()`.
Examples

data(Salamanders, package = "glmmTMB")
model <- glmmTMB::glmmTMB(
  count ~ mined + spp + (1 | site),
  family = poisson(),
  data = Salamanders
)
simulated_residuals <- performance::simulate_residuals(model)
plot(simulated_residuals)

# or
simulated_residuals <- performance::simulate_residuals(model)
result <- performance::check_residuals(simulated_residuals)
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
library(rstanarm)
library(bayestestR)
set.seed(123)
m <- suppressWarnings(stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0))
result <- point_estimate(m, centrality = "median")
result
plot(result)
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
library(rstanarm)
library(bayestestR)
set.seed(123)
m <<- suppressWarnings(stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0))
result <- p_direction(m)
plot(result)
```

## S3 method for class 'see_p_function'
plot(  
  x,  
  colors = c("black", "#1b6ca8"),  
  size_point = 1.2,  
  size_line = c(0.7, 0.9),  
  size_text = 3,  
  line_alpha = 0.15,
```
Arguments

- **x**: An object returned by `parameters::p_function()`.
- **colors**: Character vector of length two, indicating the colors (in hex-format) used when only one parameter is plotted, resp. when panels are plotted as facets.
- **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.
- **line_alpha**: Numeric value specifying alpha of lines indicating the emphasized compatibility interval levels (see `?parameters::p_function`).
- **show_labels**: Logical. If TRUE, text labels are displayed.
- **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_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.

Examples

```r
library(parameters)
model <- lm(Sepal.Length ~ Species + Sepal.Width + Petal.Length, data = iris)
result <- p_function(model)
plot(result, n_columns = 2, show_labels = FALSE)

result <- p_function(model, keep = "Sepal.Width")
plot(result)
```
plot.see_p_significance

Plot method for practical significance

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.
- **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
library(rstanarm)
library(bayestestR)
set.seed(123)
m <<- suppressWarnings(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.
- `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.
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

```r
library(rstanarm)
library(bayestestR)
set.seed(123)
m <<- suppressWarnings(stan_glm(Sepal.Length ~ Petal.Width * Species, data = iris, refresh = 0))
result <- si(m, verbose = FALSE)
result
plot(result)
```

Description

A wrapper around `patchwork` to plot multiple figures side by side on the same page.

Usage

```r
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

- `...`: Multiple 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 nu-
merals. 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 patch-
work.
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.

Details
See the patchwork documentation for more advanced control of plot layouts.

Examples
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, p3,
  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

```
## S3 method for class 'see_performance_pp_check'
print(
  x,
  size_line = 0.5,
  size_point = 2,
  size_bar = 0.7,
  size_axis_title = base_size,
  size_title = 12,
  base_size = 10,
  line_alpha = 0.15,
  style = theme_lucid,
  colors = unname(social_colors(c("green", "blue"))),
  type = c("density", "discrete_dots", "discrete_interval", "discrete_both"),
  x_limits = NULL,
  ...
)
```

```
## S3 method for class 'see_performance_pp_check'
plot(
  x,
  size_line = 0.5,
  size_point = 2,
  size_bar = 0.7,
  size_axis_title = base_size,
  size_title = 12,
  base_size = 10,
  line_alpha = 0.15,
  style = theme_lucid,
  colors = unname(social_colors(c("green", "blue"))),
  type = c("density", "discrete_dots", "discrete_interval", "discrete_both"),
  x_limits = NULL,
  ...
)
```

## Arguments

- **x**: An object.
- **size_line**: Numeric value specifying size of line geoms.
- **size_point**: Numeric specifying size of point-geoms.
- **size_bar**: Size of bar geoms.
- **base_size**, **size_axis_title**, **size_title**: Numeric value specifying size of axis and plot titles.
- **line_alpha**: Numeric value specifying alpha of lines indicating yrep.
- **style**: A ggplot2-theme.
- **colors**: Character vector of length two, indicating the colors (in hex-format) for points and line.
**type**

Plot type for the posterior predictive checks plot. Can be "density" (default), "discrete_dots", "discrete_interval" or "discrete_both" (the discrete_* options are appropriate for models with discrete - binary, integer or ordinal etc. - outcomes).

**x_limits**

Numeric vector of length 2 specifying the limits of the x-axis. If not NULL, will zoom in the x-axis to the specified limits.

**...**

Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**See Also**

See also the vignette about `check_model()`.

**Examples**

```r
library(performance)

model <- lm(Sepal.Length ~ Species * Petal.Width + Petal.Length, data = iris)
check_predictions(model)

# dot-plot style for count-models
d <- iris
d$poisson_var <- rpois(150, 1)
model <- glm(poisson_var ~ Species + Petal.Length + Petal.Width, 
data = d, 
family = poisson()
)
out <- check_predictions(model)
plot(out, type = "discrete_dots")
```

---

**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

aesthetics = "color",
...
)

scale_color_bluebrown_d(
  palette = "contrast",
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)

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

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

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

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

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

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

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

Arguments

  palette Character name of palette. Depending on the color scale, can be "full", "ice",
             "rainbow", "complement", "contrast", "light" (for dark themes), "black_first",
             full_original, or black_first_original.
  discrete Boolean indicating whether color aesthetic is discrete or not.
  reverse Boolean indicating whether the palette should be reversed.
  aesthetics A vector of names of the aesthetics that this scale should be applied to (e.g.,
               c(‘color’, ‘fill’)).
  ... 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_colorhex

Description

This function creates color scales based on palettes from https://www.color-hex.com/. This website provides a large number of user-submitted color palettes. This function downloads a requested color palette from https://www.color-hex.com/ and creates a {ggplot2} color scale from the provided hex codes.

Use `scale_color_colorhex_d` for discrete categories and `scale_color_colorhex_c` for a continuous scale.

Usage

```r
scale_color_colorhex(
    palette = 1014416,
    discrete = TRUE,
    reverse = FALSE,
    aesthetics = "color",
    ... 
)
```

```r
scale_color_colorhex_d(
    palette = 1014416,
    discrete = TRUE,
    reverse = FALSE,
    aesthetics = "color",
    ... 
)
```

```r
scale_color_colorhex_c(
    palette = 1014416,
    discrete = FALSE,
    reverse = FALSE,
    aesthetics = "color",
    ... 
)
```

```r
scale_colour_colorhex(
    palette = 1014416,
    discrete = TRUE,
    reverse = FALSE,
    aesthetics = "color",
    ... 
)
```

```r
scale_colour_colorhex_c(
    palette = 1014416,
    discrete = FALSE,
    reverse = FALSE,
    aesthetics = "color",
    ... 
)
```
scale_colour_colorhex_d(
  palette = 1014416,
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)

scale_fill_colorhex(
  palette = 1014416,
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "fill",
  ...
)

scale_fill_colorhex_d(
  palette = 1014416,
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "fill",
  ...
)

scale_fill_colorhex_c(
  palette = 1014416,
  discrete = FALSE,
  reverse = FALSE,
  aesthetics = "fill",
  ...
)

Arguments

- **palette**: The numeric code for a palette at https://www.color-hex.com/. For example, 1014416 for the Josiah color palette (number 1014416).
- **discrete**: Boolean indicating whether color aesthetic is discrete or not.
- **reverse**: Boolean indicating whether the palette should be reversed.
- **aesthetics**: A vector of names of the aesthetics that this scale should be applied to (e.g., c('color', 'fill')).
- **...**: Additional arguments to pass to colorRampPalette().

Note

The default Josiah color palette (number 1014416) is available without an internet connection. All other color palettes require an internet connection to download and access.
scale_color_flat

Examples

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

ggplot(iris, aes(x = Species, y = Sepal.Length, color = Species)) +
  geom_boxplot() +
  theme_modern() +
  scale_color_colorhex_d(palette = 1014416)

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_violin() +
  theme_modern() +
  scale_fill_colorhex_d(palette = 1014416)

ggplot(iris, aes(x = Petal.Length, y = Petal.Width, color = Sepal.Length)) +
  geom_point() +
  theme_modern() +
  scale_color_colorhex_c(palette = 1014416)
```

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

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

```r
scale_color_flat_d(
  palette = "contrast",
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)
```

```r
scale_color_flat_c(
  palette = "contrast",
  discrete = FALSE,
  ...
)
```
scale_color_flat

reverse = FALSE,
aesthetics = "color",
...

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

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

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

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

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

scale_fill_flat_c(
  palette = "contrast",
discrete = FALSE,
Arguments

- `palette` (Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast", "light" (for dark themes), "black_first", full_original, or black_first_original)
- `discrete` (Boolean indicating whether color aesthetic is discrete or not)
- `reverse` (Boolean indicating whether the palette should be reversed)
- `aesthetics` (A vector of names of the aesthetics that this scale should be applied to (e.g., c('color', 'fill')))
- ... (Additional arguments passed to discrete_scale() when discrete is TRUE or to scale_color_gradientn() when discrete is 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

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

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

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

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

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

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

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

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

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

Arguments

  palette  Character name of palette. Depending on the color scale, can be "full", "ice",
            "rainbow", "complement", "contrast", "light" (for dark themes), "black_first",
            full_original, or black_first_original.
  discrete  Boolean indicating whether color aesthetic is discrete or not.
  reverse   Boolean indicating whether the palette should be reversed.
  aesthetics A vector of names of the aesthetics that this scale should be applied to (e.g.,
              c('color', 'fill')).

...  Additional arguments to pass to colorRampPalette().

Examples

library(ggplot2)
library(see)

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

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

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

---

scale_color_metro  Metro color palette

Description

The palette based on Metro 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,
  aesthetics = "color",
  ...
)

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

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

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

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

discrete Boolean indicating whether color aesthetic is discrete or not.
reverse  
   Boolean indicating whether the palette should be reversed.

aesthetics  
   A vector of names of the aesthetics that this scale should be applied to (e.g.,
   c('color', 'fill')).

...  
   Additional arguments to pass to colorRampPalette().

Examples

library(ggplot2)
library(see)

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

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

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

scale_color_okabeito  
Okabe-Ito color palette

Description

The Okabe-Ito color palette was proposed by Okabe and Ito (2008) as a qualitative color palette
that is accessible to people with a variety of forms of color vision deficiency. In addition to being
accessible, it includes 9 vivid colors that are readily nameable and include colors that correspond to
major primary and secondary colors (e.g., red, yellow, blue).

Usage

scale_color_okabeito(
   palette = "full",
   reverse = FALSE,
   order = 1:9,
   aesthetics = "color",
   ...
)

scale_fill_okabeito(
   palette = "full",
   reverse = FALSE,
Arguments

- **palette**: Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast", "light" (for dark themes), "black_first", "full_original", or "black_first_original".
- **reverse**: Boolean indicating whether the palette should be reversed.
- **order**: A vector of numbers from 1 to 9 indicating the order of colors to use (default: 1:9)
- **aesthetics**: A vector of names of the aesthetics that this scale should be applied to (e.g., c('color', 'fill')).
- **...**: Additional arguments to pass to `colorRampPalette()`.
Details

The Okabe-Ito palette is included in the base R `grDevices::palette.colors()`. These functions make this palette easier to use with `ggplot2`.

The original Okabe-Ito palette’s “yellow” color is “#F0E442”. This color is very bright and often does not show up well on white backgrounds (see [here](#) for a discussion of this issue). Accordingly, by default, this function uses a darker more “amber” color for “yellow” (“#F5C710”). This color is the “yellow” color used in base R >4.0’s default color palette. The palettes “full” and “black_first” use this darker yellow color. For the original yellow color suggested by Okabe and Ito (“#F0E42”), use palettes “full_original” or “black_first_original”.

The Okabe-Ito palette is only available as a discrete palette. For color-accessible continuous variables, consider the [viridis palettes](#).

References


Examples

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

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

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

# for the original brighter yellow color suggested by Okabe and Ito

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

ggplot(iris, aes(x = Species, y = Sepal.Length, fill = Species)) +
  geom_violin() +
  theme_modern() +
  scale_fill_oi(order = c(1, 5, 6, 2, 4, 3, 7))
```

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,
  aesthetics = "color",
  ...
)
```

```r
scale_color_pizza_d(
  palette = "margherita",
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)
```

```r
scale_color_pizza_c(
  palette = "margherita",
  discrete = FALSE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)
```

```r
scale_colour_pizza(
  palette = "margherita",
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)
```

```r
scale_colour_pizza_c(
  palette = "margherita",
  discrete = FALSE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)
```

```r
scale_colour_pizza_d(
  palette = "margherita",
```
scale_color_pizza

```r
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)
```

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

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

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

**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.
- `aesthetics` A vector of names of the aesthetics that this scale should be applied to (e.g., `c('color', 'fill')`).
- `...` 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_color_see

```r
scale_fill_pizza_d()

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,
  aesthetics = "color",
  ...
)

scale_color_see_d(
  palette = "contrast",
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)

scale_color_see_c(
  palette = "contrast",
  discrete = FALSE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)

scale_colour_see(
  palette = "contrast",
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "color",
  ...
)
```
scale_color_see

scale_colour_see_c(
    palette = "contrast",
    discrete = FALSE,
    reverse = FALSE,
    aesthetics = "color",
    ...
)

scale_colour_see_d(
    palette = "contrast",
    discrete = TRUE,
    reverse = FALSE,
    aesthetics = "color",
    ...
)

scale_fill_see(
    palette = "contrast",
    discrete = TRUE,
    reverse = FALSE,
    aesthetics = "fill",
    ...
)

scale_fill_see_d(
    palette = "contrast",
    discrete = TRUE,
    reverse = FALSE,
    aesthetics = "fill",
    ...
)

scale_fill_see_c(
    palette = "contrast",
    discrete = FALSE,
    reverse = FALSE,
    aesthetics = "fill",
    ...
)

Arguments

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

discrete Boolean indicating whether color aesthetic is discrete or not.
scale_color_social  

reverse  
Boolean indicating whether the palette should be reversed.

aesthetics  
A vector of names of the aesthetics that this scale should be applied to (e.g., `c('color', 'fill')`).

...  
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()

```

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

```

```r
  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,
  aesthetics = "color",
  ...
)
```

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

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

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

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

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

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

scale_fill_social_d(
  palette = "complement",
  discrete = TRUE,
  reverse = FALSE,
  aesthetics = "fill",
  ...
see_colors

...)

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

Arguments

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

- **discrete**: Boolean indicating whether color aesthetic is discrete or not.

- **reverse**: Boolean indicating whether the palette should be reversed.

- **aesthetics**: A vector of names of the aesthetics that this scale should be applied to (e.g., c('color', 'fill')).

Examples

```r
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 colors.
social_colors

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 colors.

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

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,
  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 default position of legends ("none", "left", "right", "bottom", "top", "inside")
theme_blackboard

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,
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 default position of legends ("none", "left", "right", "bottom", "top", "inside")
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).
theme_modern

- `legend.position`:
  The default position of legends ("none", "left", "right", "bottom", "top", "inside")

- `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)) +
  geom_point(color = "white") +
  theme_lucid()
```

---

**Description**

A modern, sleek and elegant theme for ggplot.

**Usage**

```r
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"
)

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 default position of legends ("none", "left", "right", "bottom", "top", "inside")
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()
```

### 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"
)
```
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 default position of legends ("none", "left", "right", "bottom", "top", "inside")
- **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

cord_radar()
Examples

data <- datawizard::reshape_longer(
  aggregate(iris[-5], list(Species = iris$Species), mean),
)

ggplot(
  data,
  aes(
    x = name,
    y = value,
    color = Species,
    group = Species,
    fill = Species
  )
) +
  geom_polygon(linewidth = 1, alpha = 0.1) +
  coord_radar() +
  theme_radar()
Index

add_plot_attributes, 4
aes(), 15, 17
bayestestR::bayesfactor_models(), 6
bayestestR::bayesfactor_parameters(), 6
bayestestR::equivalence_test(), 6
bayestestR::estimate_density(), 6
bayestestR::hdi(), 6
bayestestR::p_direction(), 6
bayestestR::p_significance(), 6
bayestestR::si(), 6, 62
bayestestR::simulate_prior(), 44, 57, 58, 60
bluebrown_colors, 5
borders(), 16, 18
colorRampPalette(), 22–26, 68, 70, 75, 78, 79, 82, 85, 87
coord_radar, 5
coord_radar(), 96
data_plot, 6
effectsize::effectsize(), 6
flat_colors, 8
fortify(), 15, 17
geom_binomdensity, 9
geom_count2 (geom_point2), 12
geom_count_borderless (geom_point2), 12
geom_from_list, 10
geom_jitter2 (geom_point2), 12
geom_jitter_borderless (geom_point2), 12
geom_point2, 12
geom_point_borderless (geom_point2), 12
geom_pointrange2 (geom_point2), 12
geom_pointrange_borderless (geom_point2), 12
geom_pooljitter (geom_poolpoint), 13
geom_poolpoint, 13
geom_violindot, 15
geom_violinhalf, 17
ggplot2::geom_count(), 13
ggplot2::geom_jitter(), 13
ggplot2::geom_point(), 13
ggplot2::geom_pointrange(), 13
golden_ratio, 19
grDevices::palette.colors(), 80
key glyphs, 16, 19
layer position, 18
layer stat, 18
layer(), 16, 18, 19
material_colors, 20
metro_colors, 20
modelbased::estimate_contrasts(), 6
oi_colors (okabeito_colors), 21
okabeito_colors, 21
palette_bluebrown, 22
palette_colorhex, 22
palette_flat, 23
palette_material, 23
palette_metro, 24
palette_oj (palette_okabeito), 24
palette_okabeito, 24
palette_pizza, 25
palette_see, 26
palette_social, 26
parameters::compare_parameters(), 6
parameters::describe_distribution(), 6
parameters::model_parameters(), 6
parameters::n_clusters(), 6
parameters::n_factors(), 6
parameters::principal_components(), 6
INDEX

parameters::simulate_parameters(), 6
patchwork::plot_layout(), 63
performance::check_collinearity(), 6
performance::check_heteroscedasticity(), 7
performance::check_homogeneity(), 7
performance::check_normality(), 7
performance::check_outliers(), 7
performance::check_predictions(), 7
performance::compare_performance(), 7
performance::performance_roc(), 7
pizza_colors, 27
plot.datawizard_table
  (plot.datawizard_tables), 27
plot.datawizard_tables, 27
plot.see_bayesfactor_models, 28
plot.see_bayesfactor_parameters, 30
plot.see_check_collinearity, 31
plot.see_check_distribution, 32
plot.see_check_heteroscedasticity, 32
plot.see_check_homogeneity, 33
plot.see_check_model, 34
plot.see_check_normality, 35
plot.see_check_outliers, 37
plot.see_compare_parameters, 38
plot.see_compare_performance, 39
plot.see_effectsize_table, 40
plot.see_equivalence_test
  (plot.see_equivalence_test_effectsize), 41
plot.see_equivalence_test_effectsize, 41
plot.see_equivalence_test_lm
  (plot.see_equivalence_test_effectsize), 41
plot.see_estimate_contrasts, 42
plot.see_estimate_density, 43
plot.see_n_factors, 45
plot.see_p_direction, 57
plot.see_p_function, 58
plot.see_p_significance, 60
plot.see_parameters_brms_meta, 46
plot.see_parameters_distribution, 48
plot.see_parameters_model, 49
plot.see_parameters_pca, 51
plot.see_parameters_sem
  (plot.see_parameters_model), 49
plot.see_parameters_simulate, 52
plot.see_performance_pp_check
  (plot.see_performance_pp_check), 64
plot.see_performance_roc, 54
plot.see_performance_simses, 54
plot.see_point_estimate, 56
plot.see_rope, 61
plot.see_si, 62
plots, 63
print.see_performance_pp_check, 64
scale_color_bluebrown, 66
scale_color_bluebrown(), 22
scale_color_bluebrown_c
  (scale_color_bluebrown), 66
scale_color_bluebrown_d
  (scale_color_bluebrown), 66
scale_color_colorhex, 68
scale_color_colorhex(), 22
scale_color_colorhex_c
  (scale_color_colorhex), 68
scale_color_colorhex_d
  (scale_color_colorhex), 68
scale_color_flat, 71
scale_color_flat(), 23
scale_color_flat_c (scale_color_flat), 71
scale_color_flat_d (scale_color_flat), 71
scale_color_material, 73
scale_color_material(), 24, 25
scale_color_material_c
  (scale_color_material), 73
scale_color_material_d
  (scale_color_material), 73
scale_color_metro, 76
scale_color_metro(), 24
scale_color_metro_c
  (scale_color_metro), 76
scale_color_metro_d
  (scale_color_metro), 76
scale_color_okabeito (scale_color_okabeito), 78
scale_color_okabeito, 78
scale_color_pizza, 80
scale_color_pizza(), 25
scale_color_pizza_c
  (scale_color_pizza), 80
scale_fill_see_d(scale_color_see), 83
scale_fill_social(scale_color_social), 85
scale_fill_social_c
   (scale_color_social), 85
scale_fill_social_d
   (scale_color_social), 85
see_colors, 87
social_colors, 88

the viridis palettes, 80
theme_abyss, 89
theme_blackboard, 90
theme_lucid, 92
theme_modern, 93
theme_radar, 95
theme_radar_dark (theme_radar), 95