

# Package ‘see’

March 31, 2022

**Type** Package

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

**Version** 0.7.0

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

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

**License** GPL-3

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

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

**Depends** graphics, grDevices, R (>= 3.4), stats

**Imports** bayestestR (>= 0.11.5), datawizard (>= 0.4.0), effectsize (>= 0.6), ggplot2 (>= 3.3.5), insight (>= 0.17.0), parameters (>= 0.17.0)

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

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**RoxygenNote** 7.1.2

**NeedsCompilation** no

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

@Dom\_Makowski),  
 Indrajeet Patil [aut, ctb] (<<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** 2022-03-31 11:00:03 UTC

## R topics documented:

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

plot.see_effectsize_table . . . . .	33
plot.see_equivalence_test_effectsize . . . . .	34
plot.see_estimate_contrasts . . . . .	35
plot.see_estimate_density . . . . .	36
plot.see_hdi . . . . .	37
plot.see_n_factors . . . . .	38
plot.see_parameters_brms_meta . . . . .	39
plot.see_parameters_distribution . . . . .	41
plot.see_parameters_model . . . . .	42
plot.see_parameters_pca . . . . .	44
plot.see_parameters_simulate . . . . .	45
plot.see_performance_roc . . . . .	46
plot.see_point_estimate . . . . .	47
plot.see_p_direction . . . . .	48
plot.see_p_significance . . . . .	49
plot.see_rope . . . . .	50
plot.see_si . . . . .	52
plots . . . . .	53
print.see_performance_pp_check . . . . .	54
scale_color_bluebrown . . . . .	55
scale_color_flat . . . . .	57
scale_color_material . . . . .	59
scale_color_metro . . . . .	61
scale_color_pizza . . . . .	63
scale_color_see . . . . .	65
scale_color_social . . . . .	66
see_colors . . . . .	68
social_colors . . . . .	69
theme_abyss . . . . .	69
theme_blackboard . . . . .	71
theme_lucid . . . . .	72
theme_modern . . . . .	74
theme_radar . . . . .	76
<b>Index</b>	<b>79</b>

---

add\_plot\_attributes     *Complete figure with its 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**

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

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

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

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

p
p + add_plot_attributes(data)

## End(Not run)
```

---

bluebrown\_colors      *Extract blue-brown colors as hex codes*

---

**Description**

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

**Usage**

```
bluebrown_colors(...)
```

**Arguments**

`...` Character names of colors.

**Value**

A character vector with color-codes.

**Examples**

```
bluebrown_colors()

bluebrown_colors("blue", "brown")
```

---

coord_radar	<i>Radar coordinate system</i>
-------------	--------------------------------

---

**Description**

Add a radar coordinate system useful for radar charts.

**Usage**

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

**Arguments**

theta	variable to map angle to (x or y)
start	Offset of starting point from 12 o'clock in radians. Offset is applied clockwise or anticlockwise depending on value of direction.
direction	1, clockwise; -1, anticlockwise
...	Other arguments to be passed to ggproto.

**Examples**

```
# Create a radar/spider chart with ggplot:
if (require("poorman") && require("ggplot2")) {
  data <- iris[-5] %>%
    aggregate(list(Species = iris$Species), mean) %>%
    datawizard::reshape_longer(2:5)

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

---

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

---

## Description

`data_plot()` extracts and transforms an object for plotting, while `plot()` visualizes results of functions from different packages in [easystats-project](#). See the documentation for your object's class:

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

## Usage

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

## Arguments

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

## Details

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

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

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

## See Also

[Package-Vignettes](#)

## Examples

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

  x <- rope(model)
  plot(x)

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

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

  x <- p_direction(model)
  plot(x)

  model <- stan_glm(
    mpg ~ wt + gear + cyl + disp,
    chains = 2,
    iter = 200,
```

```
    refresh = 0,  
    data = mtcars  
  )  
  x <- equivalence_test(model)  
  plot(x)  
}  
  
## End(Not run)
```

---

**flat\_colors***Extract Flat UI colors as hex codes*

---

**Description**

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

**Usage**

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

**Arguments**

... Character names of colors.

**Value**

A character vector with color-codes.

**Examples**

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

---

**geom\_binomdensity***Add dot-densities for binary y variables*

---

**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**

```
library(ggplot2)
library(see)

data <- iris[1:100, ]

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

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

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

---

geom\_from\_list

---

*Create ggplot2 geom(s) from a list*


---

**Description**

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

**Usage**

```
geom_from_list(x, ...)
```

```
geoms_from_list(x, ...)
```

**Arguments**

**x** A list containing:

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

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

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

**Examples**

```
library(ggplot2)

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

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

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

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

```

)
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(geom = "smooth", color = "red")) +
  geom_from_list(list(aes = list(x = "Sepal.Length"), geom = "ggside::geom_xsidedensity")) +
  geom_from_list(list(geom = "ggside::scale_xsidey_continuous", breaks = NULL))

# Example 6 (ggraph) -----
if (require("tidygraph", quietly = TRUE) &&
    require("ggraph", quietly = TRUE)) {
  # Prepare graph
  nodes <- data.frame(name = c("Dom", "Mattan", "Daniel", "Brenton"))
  edges <- data.frame(
    from = c(1, 1, 1, 2, 3, 3, 4, 4, 4),
    to = c(2, 3, 4, 1, 1, 2, 1, 2, 3)
  )
  data <- tidygraph::tbl_graph(nodes = nodes, edges = edges)

  ggraph(data, layout = "kk") +
    geom_from_list(list(geom = "ggraph::geom_edge_arc")) +
    geom_from_list(list(geom = "ggraph::geom_node_point", size = 10)) +
    geom_from_list(list(geom = "ggraph::geom_node_label", aes = list(label = "name")))
}

```

---

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_jitter_borderless(...)
geom_pointrange_borderless(...)
```

### Arguments

...	Other arguments to be passed to <code>ggplot2::geom_point()</code> , <code>ggplot2::geom_jitter()</code> , <code>ggplot2::geom_pointrange()</code> , or <code>ggplot2::geom_count()</code> .
stroke	Stroke thickness.
shape	Shape of points.
size	Size of points.

### Note

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

### Examples

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

---

geom\_poolpoint

*Pool ball points*

---

### Description

Points labelled with the observation name.

**Usage**

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

**Arguments**

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

**Examples**

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

---

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

---

## Description

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

## Usage

```
geom_violindot(  
  mapping = NULL,  
  data = NULL,  
  trim = TRUE,  
  scale = 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 <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code> ).
trim	If <code>TRUE</code> (default), trim the tails of the violins to the range of the data. If <code>FALSE</code> , don't trim the tails.

scale	if "area" (default), all violins have the same area (before trimming the tails). If "count", areas are scaled proportionally to the number of observations. If "width", all violins have the same maximum width.
show.legend	logical. Should this layer be included in the legends? 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. <code>borders()</code> .
binwidth	When method is "dotdensity", this specifies maximum bin width. When method is "histodot", this specifies bin width. Defaults to 1/30 of the range of the data
position_dots	Position adjustment for dots, either as a string, or the result of a call to a position adjustment function.
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired geom/stat.
size_dots, dots_size	Size adjustment for dots.
color_dots, dots_color	Color adjustment for dots.
fill_dots, dots_fill	Fill adjustment for dots.

### Examples

```
library(ggplot2)
library(see)

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

---

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

---

### Description

Create a half-violin plot.

### Usage

```
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 <code>aes()</code> or <code>aes_()</code> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If <code>NULL</code> , the default, the data is inherited from the plot data as specified in the call to <code>ggplot()</code> . A <code>data.frame</code> , or other object, will override the plot data. All objects will be fortified to produce a data frame. See <code>fortify()</code> for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code> , and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code> ).
stat	The statistical transformation to use on the data for this layer, as a string.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
trim	If <code>TRUE</code> (default), trim the tails of the violins to the range of the data. If <code>FALSE</code> , don't trim the tails.
flip	Should the half-violin plot switch directions? By default, this is <code>FALSE</code> and all half-violin geoms will have the flat-side on facing leftward. If <code>flip = TRUE</code> , 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? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <code>borders()</code> .
...	Other arguments passed on to <code>layer()</code> . These are often aesthetics, used to set an aesthetic to a fixed value, like <code>colour = "red"</code> or <code>size = 3</code> . They may also be parameters to the paired geom/stat.

## Examples

```
library(ggplot2)
library(see)

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

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

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

---

golden\_ratio

*Golden Ratio*

---

## Description

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

## Usage

```
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

```
golden_ratio()
golden_ratio(10)
```

---

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

---

**Description**

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

**Usage**

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

**Arguments**

... Character names of colors.

**Value**

A character vector with color-codes.

**Examples**

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

---

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

---

**Description**

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

**Usage**

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

**Arguments**

... Character names of colors.

**Value**

A character vector with color-codes.

**Examples**

```
metro_colors()

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

---

palette\_bluebrown      *Blue-brown design color palette*

---

**Description**

The palette based on blue-brown colors.

**Usage**

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

**Arguments**

palette	Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

**Details**

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

---

palette\_flat      *Flat UI color palette*

---

**Description**

The palette based on **Flat UI**.

**Usage**

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

**Arguments**

palette	Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

**Details**

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

---

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

---

**Description**

The palette based on [material design colors](#).

**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" or "light" (for dark themes).
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <a href="#">colorRampPalette()</a> .

**Details**

This function is usually not called directly, but from within [scale\\_color\\_material\(\)](#).

---

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

---

**Description**

The palette based on [Metro colors](#).

**Usage**

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

**Arguments**

palette	Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <a href="#">colorRampPalette()</a> .

**Details**

This function is usually not called directly, but from within [scale\\_color\\_metro\(\)](#).

---

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

---

**Description**

The palette based on authentic neapolitan pizzas.

**Usage**

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

**Arguments**

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

**Details**

This function is usually not called directly, but from within [scale\\_color\\_pizza\(\)](#).

---

palette_see	<i>See design color palette</i>
-------------	---------------------------------

---

**Description**

See design color palette

**Usage**

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

**Arguments**

palette	Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <a href="#">colorRampPalette()</a> .

**Details**

This function is usually not called directly, but from within [scale\\_color\\_see\(\)](#).

---

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

---

**Description**

The palette based **Social colors**.

**Usage**

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

**Arguments**

palette	Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <code>colorRampPalette()</code> .

**Details**

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

---

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

---

**Description**

Extract pizza colors as hex codes

**Usage**

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

**Arguments**

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

**Value**

A character vector with color-codes.

---

```
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

<code>x</code>	An object.
<code>n_pies</code>	Number of pies.
<code>value</code>	What value to display.
<code>sort</code>	The behavior of this argument depends on the plotting contexts. <ul style="list-style-type: none"> <li>• <i>Plotting model parameters</i>: If <code>NULL</code>, coefficients are plotted in the order as they appear in the summary. Setting <code>sort = "ascending"</code> or <code>sort = "descending"</code> sorts coefficients in ascending or descending order, respectively. Setting <code>sort = TRUE</code> is the same as <code>sort = "ascending"</code>.</li> <li>• <i>Plotting Bayes factors</i>: Sort pie-slices by posterior probability (descending)?</li> </ul>
<code>log</code>	Logical that decides whether to display log-transformed Bayes factors.
<code>prior_odds</code>	An optional vector of prior odds for the models. See <code>BayesFactor::priorOdds</code> . As the size of the pizza slices corresponds to posterior probability (which is a function of prior probability and the Bayes Factor), custom <code>prior_odds</code> will change the slices' size.
<code>...</code>	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**

```

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

```

**Arguments**

<code>x</code>	An object.
<code>size_point</code>	Numeric specifying size of point-geoms.
<code>rope_color</code>	Character specifying color of ROPE ribbon.
<code>rope_alpha</code>	Numeric specifying transparency level of ROPE ribbon.
<code>show_intercept</code>	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.
<code>...</code>	Arguments passed to or from other methods.

**Value**

A ggplot2-object.

---

```
plot.see_check_collinearity
```

*Plot method for multicollinearity checks*

---

**Description**

The plot() method for the performance::check\_collinearity() function.

**Usage**

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

**Arguments**

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

**Value**

A ggplot2-object.

**Examples**

```
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**

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

**Arguments**

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

**Value**

A `ggplot2`-object.

**Examples**

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

---

`plot.see_check_heteroscedasticity`*Plot method for (non-)constant error variance checks*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_check\_homogeneity

*Plot method for homogeneity of variances checks*

---

**Description**

The plot() method for the performance::check\_homogeneity() function.

**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**

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

---

```
plot.see_check_normality
```

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

---

**Description**

The plot() method for the performance::check\_normality() function.

**Usage**

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

**Arguments**

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

**Value**

A ggplot2-object.

**Examples**

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

---

```
plot.see_check_outliers
```

*Plot method for checking outliers*

---

**Description**

The plot() method for the performance::check\_outliers() function.

**Usage**

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

**Arguments**

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

```
## 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. <ul style="list-style-type: none"> <li>• <i>Plotting model parameters</i>: 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".</li> <li>• <i>Plotting Bayes factors</i>: Sort pie-slices by posterior probability (descending)?</li> </ul>
n_columns	For models with multiple components (like fixed and random, count and zero-inflated), defines the number of columns for the panel-layout. If NULL, a single, integrated plot is shown.
show_labels	Logical. If TRUE, text labels are displayed.
...	Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_compare\_performance

*Plot method for comparing model performances*

---

**Description**

The plot() method for the performance::compare\_performance() function.

**Usage**

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



**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**

```
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**

```
## 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**

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

---

```
plot.see_equivalence_test_effectsize
```

*Plot method for (conditional) equivalence testing*

---

## Description

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

## Usage

```
## 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

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

n_columns	For models with multiple components (like fixed and random, count and zero-inflated), defines the number of columns for the panel-layout. If NULL, a single, integrated plot is shown.
size_point	Numeric specifying size of point-geoms.

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_estimate\_contrasts

*Plot method for estimating contrasts*

---

**Description**

The plot() method for the modelbased::estimate\_contrasts() function.

**Usage**

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

**Arguments**

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

**Value**

A ggplot2-object.

**Examples**

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

---

```
plot.see_estimate_density
```

*Plot method for density estimation of posterior samples*

---

## Description

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

## Usage

```
## 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

<code>x</code>	An object.
<code>stack</code>	Logical. If TRUE, densities are plotted as stacked lines. Else, densities are plotted for each parameter among each other.
<code>show_intercept</code>	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.
<code>n_columns</code>	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.
<code>priors</code>	Logical. If TRUE, prior distributions are simulated (using <code>bayestestR::simulate_prior()</code> ) and added to the plot.
<code>priors_alpha</code>	Numeric value specifying alpha for the prior distributions.
<code>posteriors_alpha</code>	Numeric value specifying alpha for the posterior distributions.
<code>size_line</code>	Numeric value specifying size of line geoms.
<code>size_point</code>	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 .95.
...	Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_hdi

*Plot method for uncertainty or credible intervals*

---

**Description**

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

**Usage**

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

**Arguments**

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

show_intercept	Logical, if TRUE, the intercept-parameter is included in the plot. By default, it is hidden because in many cases the intercept-parameter has a posterior distribution on a very different location, so density curves of posterior distributions for other parameters are hardly visible.
show_zero	Logical. If TRUE, will add a vertical (dotted) line at 0.
show_title	Logical. If TRUE, will show the title of the plot.
n_columns	For models with multiple components (like fixed and random, count and zero-inflated), defines the number of columns for the panel-layout. If NULL, a single, integrated plot is shown.
...	Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_n\_factors      *Plot method for numbers of clusters to extract or factors to retain*

---

**Description**

The plot() method for the parameters::n\_factors() and parameters::n\_clusters()

**Usage**

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

**Arguments**

x	An object.
data	The original data used to create this object. Can be a statistical model.
type	Character vector, indicating the type of plot.
size	Depending on type, a numeric value specifying size of bars, lines, or segments.
...	Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_parameters\_brms\_meta

*Plot method for Model Parameters from Bayesian Meta-Analysis*

---

**Description**

The plot() method for the parameters::model\_parameters() function when used with brms-meta-analysis models.

**Usage**

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

**Arguments**

x	An object.
size_point	Numeric specifying size of point-geoms.
size_line	Numeric value specifying size of line geoms.
size_text	Numeric value specifying size of text labels.
posteriors_alpha	Numeric value specifying alpha for the posterior distributions.
rope_alpha	Numeric specifying transparency level of ROPE ribbon.

rope_color	Character specifying color of ROPE ribbon.
normalize_height	Logical. If TRUE, height of mcmc-areas is "normalized", to avoid overlap. In certain cases when the range of a posterior distribution is narrow for some parameters, this may result in very flat mcmc-areas. In such cases, set <code>normalize_height = FALSE</code> .
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

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

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

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



```

}

## End(Not run)

```

---

```

plot.see_parameters_distribution
Plot method for describing distributions of vectors

```

---

## Description

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

## Usage

```

## 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

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

---

plot.see\_parameters\_model

*Plot method for model parameters*

---

**Description**

The plot() method for the parameters::model\_parameters() function.

**Usage**

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

## 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. <ul style="list-style-type: none"> <li>• <i>Plotting model parameters</i>: 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".</li> <li>• <i>Plotting Bayes factors</i>: Sort pie-slices by posterior probability (descending)?</li> </ul>
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. <b>metafor</b> ).
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.
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**

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

---

```
plot.see_parameters_simulate
```

*Plot method for simulated model parameters*

---

**Description**

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

**Usage**

```
## 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**

<code>x</code>	An object.
<code>data</code>	The original data used to create this object. Can be a statistical model.
<code>stack</code>	Logical. If TRUE, densities are plotted as stacked lines. Else, densities are plotted for each parameter among each other.
<code>show_intercept</code>	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.
<code>n_columns</code>	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 <code>normalize_height = FALSE</code> .
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 .95.
...	Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_performance\_roc  
*Plot method for ROC curves*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

A ggplot2-object.

**Examples**

```
library(performance)
data(iris)
set.seed(123)
iris$y <- rbinom(nrow(iris), size = 1, .3)

folds <- sample(nrow(iris), size = nrow(iris) / 8, replace = FALSE)
test_data <- iris[folds, ]
train_data <- iris[-folds, ]

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

---

plot.see\_point\_estimate

*Plot method for point estimates of posterior samples*

---

**Description**

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

**Usage**

```
## 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**

<code>x</code>	An object.
<code>data</code>	The original data used to create this object. Can be a statistical model.
<code>size_point</code>	Numeric specifying size of point-geoms.
<code>size_text</code>	Numeric value specifying size of text labels.
<code>panel</code>	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 <code>bayestestR::simulate_prior()</code> ) 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**

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

---

plot.see\_p\_direction *Plot method for probability of direction*

---

**Description**

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

**Usage**

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



**Arguments**

x	An object.
data	The original data used to create this object. Can be a statistical model.
show_intercept	Logical, if TRUE, the intercept-parameter is included in the plot. By default, it is hidden because in many cases the intercept-parameter has a posterior distribution on a very different location, so density curves of posterior distributions for other parameters are hardly visible.
priors	Logical. If TRUE, prior distributions are simulated (using <code>bayestestR::simulate_prior()</code> ) and added to the plot.
priors_alpha	Numeric value specifying alpha for the prior distributions.
n_columns	For models with multiple components (like fixed and random, count and zero-inflated), defines the number of columns for the panel-layout. If NULL, a single, integrated plot is shown.
...	Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_p\_significance

*Plot method for practical significance*

---

**Description**

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

**Usage**

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

```

  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 <code>bayestestR::simulate_prior()</code> ) and added to the plot.
priors_alpha	Numeric value specifying alpha for the prior distributions.
n_columns	For models with multiple components (like fixed and random, count and zero-inflated), defines the number of columns for the panel-layout. If NULL, a single, integrated plot is shown.
...	Arguments passed to or from other methods.

### Value

A ggplot2-object.

### Examples

```

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

```

---

plot.see\_rope

*Plot method for Region of Practical Equivalence*

---

### Description

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

**Usage**

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

**Arguments**

x	An object.
data	The original data used to create this object. Can be a statistical model.
rope_alpha	Numeric specifying transparency level of ROPE ribbon.
rope_color	Character specifying color of ROPE ribbon.
show_intercept	Logical, if TRUE, the intercept-parameter is included in the plot. By default, it is hidden because in many cases the intercept-parameter has a posterior distribution on a very different location, so density curves of posterior distributions for other parameters are hardly visible.
n_columns	For models with multiple components (like fixed and random, count and zero-inflated), defines the number of columns for the panel-layout. If NULL, a single, integrated plot is shown.
...	Arguments passed to or from other methods.

**Value**

A ggplot2-object.

**Examples**

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

---

plot.see\_si *Plot method for support intervals*

---

### Description

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

### Usage

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

### Arguments

<code>x</code>	An object.
<code>si_color</code>	Character specifying color of SI ribbon.
<code>si_alpha</code>	Numeric value specifying Transparency level of SI ribbon.
<code>show_intercept</code>	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.
<code>support_only</code>	Logical. Decides whether to plot only the support data, or show the "raw" prior and posterior distributions? Only applies when plotting <code>bayestestR::si()</code> .
<code>...</code>	Arguments passed to or from other methods.

### Value

A `ggplot2`-object.

### Examples

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

---

plots *Multiple plots side by side*

---

### Description

A wrapper around *patchwork* to plot multiple figures side by side on the same page. See [the \*patchwork\* documentation](#) for more advanced control of plot layouts.

### Usage

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

### Arguments

...	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 <a href="#">patchwork::plot_layout()</a> for details.
tags	Add tags to your subfigures. Can be NULL to omit (default) or a character vector containing tags for each plot. Automatic tags can also be generated with '1' for Arabic numerals, 'A' for uppercase Latin letters, 'a' for lowercase Latin letters, 'I' for uppercase Roman numerals, and 'i' for lowercase Roman numerals. For backwards compatibility, can also be FALSE (equivalent to NULL), NA (equivalent to NULL), or TRUE (equivalent to 'A').
tag_prefix, tag_suffix	Text strings that should appear before or after the tag.
tag_sep	Text string giving the separator to use between different tag levels.
title, subtitle, caption	Text strings to use for the various plot annotations to add to the composed patchwork.

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

### Examples

```
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,
  line_alpha = 0.15,
  size_bar = 0.7,
  style = theme_lucid,
  colors = unname(social_colors(c("green", "blue"))),
  ...
)

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

```

    line_alpha = 0.15,
    size_bar = 0.7,
    style = theme_lucid,
    colors = unname(social_colors(c("green", "blue"))),
    ...
  )

```

### Arguments

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

### Value

A ggplot2-object.

### Examples

```

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

```

---

scale\_color\_bluebrown *Blue-brown color palette*

---

### Description

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

### Usage

```

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

```

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

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

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

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

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

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

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



```

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

```

### Arguments

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

### Examples

```

library(ggplot2)
library(see)

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

```

---

scale\_color\_flat      *Flat UI color palette*

---

### Description

The palette based on **Flat UI**. Use `scale_color_flat_d` for *discrete* categories and `scale_color_flat_c` for a *continuous* scale.

### Usage

```

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

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

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

```

```

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

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

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

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

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

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

```

### Arguments

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

### Examples

```

library(ggplot2)
library(see)

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

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

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

```

---

scale\_color\_material *Material design color palette*

---

### Description

The palette based on **material design colors**. Use `scale_color_material_d()` for *discrete* categories and `scale_color_material_c()` for a *continuous* scale.

### Usage

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

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

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

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

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

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

```

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

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

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

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

```

### Arguments

palette	Character name of palette. Depending on the color scale, can be "full", "ice", "rainbow", "complement", "contrast" or "light" (for dark themes).
discrete	Boolean indicating whether color aesthetic is discrete or not.
reverse	Boolean indicating whether the palette should be reversed.
...	Additional arguments to pass to <a href="#">colorRampPalette()</a> .

### 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,  
  ...  
)
```

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

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

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

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

```

    ...
  )

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

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

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

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

```

### Arguments

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

### Examples

```

library(ggplot2)
library(see)

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

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

```

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

---

scale\_color\_pizza      *Pizza color palette*

---

### Description

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

### Usage

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

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

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

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

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

```

    ...
  )

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

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

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

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

```

### Arguments

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

### Examples

```

library(ggplot2)
library(see)

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

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

```



---

scale\_color\_see      *See color palette*

---

### Description

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

### Usage

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

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

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

### Arguments

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

### Examples

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

```

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

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

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

```

---

scale\_color\_social      *Social color palette*

---

### Description

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

### Usage

```

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

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

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

scale_colour_social(
  palette = "complement",

```

```
    discrete = TRUE,  
    reverse = FALSE,  
    ...  
  )  
  
scale_colour_social_c(  
  palette = "complement",  
  discrete = FALSE,  
  reverse = FALSE,  
  ...  
)  
  
scale_colour_social_d(  
  palette = "complement",  
  discrete = TRUE,  
  reverse = FALSE,  
  ...  
)  
  
scale_fill_social(  
  palette = "complement",  
  discrete = TRUE,  
  reverse = FALSE,  
  ...  
)  
  
scale_fill_social_d(  
  palette = "complement",  
  discrete = TRUE,  
  reverse = FALSE,  
  ...  
)  
  
scale_fill_social_c(  
  palette = "complement",  
  discrete = FALSE,  
  reverse = FALSE,  
  ...  
)
```

### Arguments

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

## Examples

```
library(ggplot2)
library(see)

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

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

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

---

see\_colors

*Extract See colors as hex codes*

---

## Description

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

## Usage

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

## Arguments

... Character names of colors.

## Value

A character vector with color-codes.

## Examples

```
see_colors()

see_colors("indigo", "lime")
```

---

social_colors	<i>Extract Social colors as hex codes</i>
---------------	---

---

**Description**

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

**Usage**

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

**Arguments**

... Character names of colors.

**Value**

A character vector with color-codes.

**Examples**

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

---

theme_abyss	<i>Abyss theme</i>
-------------	--------------------

---

**Description**

A deep dark blue theme for ggplot.

**Usage**

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

```

axis.title.size = 13,
axis.title.face = "plain",
axis.text.size = 12,
axis.text.angle = NULL,
tags.size = 15,
tags.face = "bold"
)

```

## Arguments

`base_size` base font size, given in pts.

`base_family` base font family

`plot.title.size` Title size in pts. Can be "none".

`plot.title.face` Title font face ("plain", "italic", "bold", "bold.italic").

`plot.title.space` Title spacing.

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

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

`axis.title.space` Axis title spacing.

`legend.title.size` Legend elements text size in pts.

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

`axis.title.size` Axis title text size in pts.

`axis.title.face` Axis font face ("plain", "italic", "bold", "bold.italic").

`axis.text.size` Axis text size in pts.

`axis.text.angle` Rotate the x axis labels.

`tags.size` Tags text size in pts.

`tags.face` Tags font face ("plain", "italic", "bold", "bold.italic").

## Examples

```
library(ggplot2)
library(see)

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

---

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

---

## Description

A modern, sleek and dark theme for ggplot.

## Usage

```
theme_blackboard(
  base_size = 11,
  base_family = "",
  plot.title.size = 15,
  plot.title.face = "plain",
  plot.title.space = 20,
  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.

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

## Examples

```
library(ggplot2)
library(see)

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

---

theme\_lucid

*Lucid theme*

---

## Description

A light, clear theme for ggplot.



**Usage**

```

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

```

**Arguments**

base_size	base font size, given in pts.
base_family	base font family
plot.title.size	Title size in pts. Can be "none".
plot.title.face	Title font face ("plain", "italic", "bold", "bold.italic").
plot.title.space	Title spacing.
plot.title.position	Alignment of the plot title/subtitle and caption. The setting for plot.title.position applies to both the title and the subtitle. A value of "panel" (the default) means that titles and/or caption are aligned to the plot panels. A value of "plot" means that titles and/or caption are aligned to the entire plot (minus any space for margins and plot tag).
legend.position	the position of legends ("none", "left", "right", "bottom", "top", or two-element numeric vector)
axis.title.space	Axis title spacing.
legend.title.size	Legend elements text size in pts.
legend.text.size	Legend elements text size in pts. Can be "none".
axis.title.size	Axis title text size in pts.

`axis.title.face` Axis font face ("plain", "italic", "bold", "bold.italic").

`axis.text.size` Axis text size in pts.

`axis.text.angle` Rotate the x axis labels.

`tags.size` Tags text size in pts.

`tags.face` Tags font face ("plain", "italic", "bold", "bold.italic").

### Examples

```
library(ggplot2)
library(see)

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

---

theme\_modern

*The easystats' minimal theme*

---

### Description

A modern, sleek and elegant theme for ggplot.

### Usage

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

**Arguments**

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

**Examples**

```
library(ggplot2)
library(see)

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

---

`theme_radar`*Themes for radar plots*

---

### Description

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

### Usage

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

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

**Arguments**

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

**See Also**

[coord\\_radar\(\)](#)

**Examples**

```
if (require("ggplot2") && require("poorman")) {
  data <- iris[-5] %>%
    aggregate(list(Species = iris$Species), mean) %>%
    datawizard::reshape_longer(2:5)
```

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

# Index

`add_plot_attributes`, 3  
`aes()`, 15, 17  
`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, 52  
`bayestestR::simulate_prior()`, 36, 48–50  
`bluebrown_colors`, 4  
`borders()`, 16, 17

`colorRampPalette()`, 20–23, 57, 60, 62, 64, 65, 67  
`coord_radar`, 5  
`coord_radar()`, 77

`data_plot`, 6

`effectsize::effectsize()`, 6

`flat_colors`, 8  
`fortify()`, 15, 17

`geom_binomdensity`, 8  
`geom_count2` (`geom_point2`), 12  
`geom_count_borderless` (`geom_point2`), 12  
`geom_from_list`, 9  
`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`, 16  
`geoms_from_list` (`geom_from_list`), 9  
`ggplot()`, 15, 17  
`ggplot2::geom_count()`, 13  
`ggplot2::geom_jitter()`, 13  
`ggplot2::geom_point()`, 13  
`ggplot2::geom_pointrange()`, 13  
`golden_ratio`, 18

`layer()`, 16, 17

`material_colors`, 19  
`metro_colors`, 19  
`modelbased::estimate_contrasts()`, 6

`palette_bluebrown`, 20  
`palette_flat`, 20  
`palette_material`, 21  
`palette_metro`, 21  
`palette_pizza`, 22  
`palette_see`, 22  
`palette_social`, 23  
`parameters::compare_parameters()`, 6  
`parameters::describe_distribution()`, 6  
`parameters::model_parameters()`, 6  
`parameters::n_clusters()`, 6  
`parameters::n_factors()`, 6  
`parameters::principal_components()`, 6  
`parameters::simulate_parameters()`, 6  
`patchwork::plot_layout()`, 53  
`performance::check_collinearity()`, 6  
`performance::check_heteroscedasticity()`, 6  
`performance::check_homogeneity()`, 6  
`performance::check_normality()`, 6  
`performance::check_outliers()`, 6  
`performance::check_posterior_predictions()`, 6

- performance::compare\_performance(), 6
- performance::performance\_roc(), 6
- pizza\_colors, 23
- plot.see\_bayesfactor\_models, 24
- plot.see\_bayesfactor\_parameters, 25
- plot.see\_check\_collinearity, 26
- plot.see\_check\_distribution, 27
- plot.see\_check\_heteroscedasticity, 27
- plot.see\_check\_homogeneity, 28
- plot.see\_check\_normality, 29
- plot.see\_check\_outliers, 30
- plot.see\_compare\_parameters, 31
- plot.see\_compare\_performance, 32
- plot.see\_effectsize\_table, 33
- plot.see\_equivalence\_test
  - (plot.see\_equivalence\_test\_effectsize), 34
- plot.see\_equivalence\_test\_effectsize, 34
- plot.see\_equivalence\_test\_lm
  - (plot.see\_equivalence\_test\_effectsize), 34
- plot.see\_estimate\_contrasts, 35
- plot.see\_estimate\_density, 36
- plot.see\_hdi, 37
- plot.see\_n\_factors, 38
- plot.see\_p\_direction, 48
- plot.see\_p\_significance, 49
- plot.see\_parameters\_brms\_meta, 39
- plot.see\_parameters\_distribution, 41
- plot.see\_parameters\_model, 42
- plot.see\_parameters\_pca, 44
- plot.see\_parameters\_sem
  - (plot.see\_parameters\_model), 42
- plot.see\_parameters\_simulate, 45
- plot.see\_performance\_pp\_check
  - (print.see\_performance\_pp\_check), 54
- plot.see\_performance\_roc, 46
- plot.see\_point\_estimate, 47
- plot.see\_rope, 50
- plot.see\_si, 52
- plots, 53
- print.see\_performance\_pp\_check, 54
  
- scale\_color\_bluebrown, 55
- scale\_color\_bluebrown(), 20
- scale\_color\_bluebrown\_c
  - (scale\_color\_bluebrown), 55
- scale\_color\_bluebrown\_d
  - (scale\_color\_bluebrown), 55
- scale\_color\_flat, 57
- scale\_color\_flat(), 20
- scale\_color\_flat\_c (scale\_color\_flat), 57
- scale\_color\_flat\_d (scale\_color\_flat), 57
- scale\_color\_material, 59
- scale\_color\_material(), 21
- scale\_color\_material\_c
  - (scale\_color\_material), 59
- scale\_color\_material\_d
  - (scale\_color\_material), 59
- scale\_color\_metro, 61
- scale\_color\_metro(), 21
- scale\_color\_metro\_c
  - (scale\_color\_metro), 61
- scale\_color\_metro\_d
  - (scale\_color\_metro), 61
- scale\_color\_pizza, 63
- scale\_color\_pizza(), 22
- scale\_color\_pizza\_c
  - (scale\_color\_pizza), 63
- scale\_color\_pizza\_d
  - (scale\_color\_pizza), 63
- scale\_color\_see, 65
- scale\_color\_see(), 22
- scale\_color\_see\_c (scale\_color\_see), 65
- scale\_color\_see\_d (scale\_color\_see), 65
- scale\_color\_social, 66
- scale\_color\_social(), 23
- scale\_color\_social\_c
  - (scale\_color\_social), 66
- scale\_color\_social\_d
  - (scale\_color\_social), 66
- scale\_colour\_bluebrown
  - (scale\_color\_bluebrown), 55
- scale\_colour\_bluebrown\_c
  - (scale\_color\_bluebrown), 55
- scale\_colour\_bluebrown\_d
  - (scale\_color\_bluebrown), 55
- scale\_colour\_flat (scale\_color\_flat), 57
- scale\_colour\_flat\_c (scale\_color\_flat), 57
- scale\_colour\_flat\_d (scale\_color\_flat), 57
- scale\_colour\_material



- (scale\_color\_material), 59
- scale\_colour\_material\_c
  - (scale\_color\_material), 59
- scale\_colour\_material\_d
  - (scale\_color\_material), 59
- scale\_colour\_metro (scale\_color\_metro), 61
- scale\_colour\_metro\_c
  - (scale\_color\_metro), 61
- scale\_colour\_metro\_d
  - (scale\_color\_metro), 61
- scale\_colour\_pizza (scale\_color\_pizza), 63
- scale\_colour\_pizza\_c
  - (scale\_color\_pizza), 63
- scale\_colour\_pizza\_d
  - (scale\_color\_pizza), 63
- scale\_colour\_see (scale\_color\_see), 65
- scale\_colour\_see\_c (scale\_color\_see), 65
- scale\_colour\_see\_d (scale\_color\_see), 65
- scale\_colour\_social
  - (scale\_color\_social), 66
- scale\_colour\_social\_c
  - (scale\_color\_social), 66
- scale\_colour\_social\_d
  - (scale\_color\_social), 66
- scale\_fill\_bluebrown
  - (scale\_color\_bluebrown), 55
- scale\_fill\_bluebrown\_c
  - (scale\_color\_bluebrown), 55
- scale\_fill\_bluebrown\_d
  - (scale\_color\_bluebrown), 55
- scale\_fill\_flat (scale\_color\_flat), 57
- scale\_fill\_flat\_c (scale\_color\_flat), 57
- scale\_fill\_flat\_d (scale\_color\_flat), 57
- scale\_fill\_material
  - (scale\_color\_material), 59
- scale\_fill\_material\_c
  - (scale\_color\_material), 59
- scale\_fill\_material\_d
  - (scale\_color\_material), 59
- scale\_fill\_metro (scale\_color\_metro), 61
- scale\_fill\_metro\_c (scale\_color\_metro), 61
- scale\_fill\_metro\_d (scale\_color\_metro), 61
- scale\_fill\_pizza (scale\_color\_pizza), 63
- scale\_fill\_pizza\_c (scale\_color\_pizza), 63
- scale\_fill\_pizza\_d (scale\_color\_pizza), 63
- scale\_fill\_see (scale\_color\_see), 65
- scale\_fill\_see\_c (scale\_color\_see), 65
- scale\_fill\_see\_d (scale\_color\_see), 65
- scale\_fill\_social (scale\_color\_social), 66
- scale\_fill\_social\_c
  - (scale\_color\_social), 66
- scale\_fill\_social\_d
  - (scale\_color\_social), 66
- see\_colors, 68
- social\_colors, 69
- theme\_abbyss, 69
- theme\_blackboard, 71
- theme\_lucid, 72
- theme\_modern, 74
- theme\_radar, 76
- theme\_radar\_dark (theme\_radar), 76