Package 'plutor'

October 27, 2023

Title Useful Functions for Visualization

Version 0.1.0

Description In ancient Roman mythology, 'Pluto' was the ruler of the underworld and presides over the afterlife. 'Pluto' was frequently conflated with 'Plutus', the god of wealth, because mineral wealth was found underground. When plotting with R, you try once, twice, practice again and again, and finally you get a pretty figure you want. It's a 'plot tour', a tour about repetition and reward. Hope 'plutor' helps you on the tour!

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Imports baizer, cli, dplyr, ggh4x, ggplot2, ggsci, grDevices, grid, magrittr, methods, purrr, RColorBrewer, repr, rlang, scales, stats, stringr, tibble, tidyr, utils

Suggests covr, svglite, testthat (>= 3.0.0), vdiffr

Config/testthat/edition 3

Depends R (>= 2.10)

LazyData true

URL https://github.com/william-swl/plutor

BugReports https://github.com/william-swl/plutor/issues

NeedsCompilation no

Author William Song [aut, cre]

Maintainer William Song <william_swl@163.com>

Repository CRAN

Date/Publication 2023-10-27 08:00:02 UTC

R topics documented:

`assign_colors` *assign colors by a column in a tibble, for the convenience to use* `scale_color_identity()`

Description

assign colors by a column in a tibble, for the convenience to use `scale_color_identity()`

Usage

```
assign_colors(df, by, colors = sci_colors("npg", 10), na = "#F5F5F5")
```

Arguments

`df`	tibble
`by`	assign colors according to this column
`colors`	a vector of color values
`na`	if colors are not enough, fill na values

Value

tibble

Examples

```
assign_colors(mini_diamond, cut, colors = sci_colors("nejm", 8))
```

`bioletter_colors` *colors of nucleotides and amino acids*

Description

colors of nucleotides and amino acids

Usage

```
bioletter_colors
```

Format

`bioletter_colors`**:**
colors for biological letters like amino acids or nucleotides

Source

according to the print format

brewer_colors *select colors from* RColorBrewer *package presets*

Description

select colors from RColorBrewer package presets

Usage

```
brewer_colors(name, n = 3, ...)
```

Arguments

name	presets name
n	number of colors
...	other arguments of RColorBrewer::brewer.pal

Value

colors

Examples

```
brewer_colors("Blues", 5)
```

canvas_size *width and height of built-in canvas*

Description

width and height of built-in canvas

Usage

```
canvas_size
```

Format

canvas_size:
canvas sizes list

Source

according to the print format

cm2inch *trans cm to inch*

Description

trans cm to inch

Usage

```
cm2inch(x)
```

Arguments

x	cm value

Value

inch value

Examples

```
cm2inch(1)
```

cm2pt *trans cm to pt*

Description

trans cm to pt

Usage

```
cm2pt(x)
```

Arguments

x	cm value

Value

pt value

Examples

```
cm2pt(1)
```

extract_compare *extract the result of* geom_compare *from a* ggplot *object*

Description

extract the result of geom_compare from a ggplot object

Usage

```
extract_compare(p)
```

Arguments

p	ggplot object

Value

compare tibble

geom2trace.GeomCompare *geom2trace.GeomCompare*

Description

geom2trace.GeomCompare

Usage

```
geom2trace.GeomCompare(data, params, plot)
```

Arguments

data, params, plot	params

Value

no return value

`geom2trace.GeomDescribe` *geom2trace.GeomDescribe*

Description

geom2trace.GeomDescribe

Usage

```
geom2trace.GeomDescribe(data, params, plot)
```

Arguments

`data, params, plot`
params

Value

no return value

`GeomCompare` *GeomCompare*

Description

GeomCompare

Usage

```
GeomCompare
```

Format

An object of class `GeomCompare` (inherits from `Geom`, `ggproto`, gg) of length 6.

GeomDescribe *GeomDescribe*

Description

GeomDescribe

Usage

```
GeomDescribe
```

Format

An object of class `GeomDescribe` (inherits from `Geom`, `ggproto`, `gg`) of length 5.

geom_compare *add p value and fold change on a plot*

Description

add p value and fold change on a plot

Usage

```
geom_compare(
  mapping = NULL,
  data = NULL,
  stat = "compare",
  position = "identity",
  ...,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  lab_pos = NULL,
  step_increase = 0.1,
  tip_length = 0.02,
  lineend = "round",
  cp_label = c("psymbol"),
  ns_lineheight_just = 0.2,
  ignore_ns = FALSE,
  fc_method = NULL,
  comparisons = NULL,
  paired = FALSE,
  alternative = "two.sided",
  test_method = "wilcoxon",
```

```
  ns_symbol = "NS",
  cp_ref = NULL,
  cp_inline = FALSE,
  brackets_widen = 0,
  fc_digits = 2,
  cp_result = NULL,
  cp_manual = NULL
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
...	Other arguments passed on to ggplot2::geom_segment().
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
lab_pos	position of the label brackets
step_increase	the increase height for next bracket, a ratio according to the whole panel height
tip_length	the length for tips at the ends of the brackets, a ratio according to the whole panel height
lineend	Line end style (round, butt, square).
cp_label	which values will be add on the plot, a character vector with some of psymbol, p, right_deno_fc, left in it. If comparisons is assigned, you can also include fc1, fc2

ns_lineheight_just
: if show psymbol in the label, justify the NS labels to make the lineheights look balanced

ignore_ns
: if TRUE will ignore all label items if p >= 0.05, or you can assign a character vector like cp_label to ignore some items of the label

fc_method
: fold change method, default is mean. If you use log10 or log2 axis, default is geom_mean.

comparisons
: a list of two-element vector, to assign the comparisons should be performed

paired
: paired test or not, FALSE as default. If TRUE, you should use mapping=aes(paired_by=col) to indicate pairs by an extra column

alternative
: one of two.sided, greater, less

test_method
: wilcoxon as default, one of wilcoxon, t

ns_symbol
: the symbol of non-significant, NS as default

cp_ref
: reference item, the others will be compared with it

cp_inline
: draw in line or not, default is FALSE

brackets_widen
: widen the brackets, can be a negative value

fc_digits
: fold change digits

cp_result
: comparation result tibble

cp_manual
: manual comparisons table, please refer to extract_compare()

Value

ggplot object

---

geom_describe *Description values plot*

Description

The describe geom is used to create description values plot, including center symbol and error symbol. The center symbol can be mean, median or other custom functions, the error symbol can be sd, quantile or other custom functions.

Usage

```
geom_describe(
  mapping = NULL,
  data = NULL,
  stat = "describe",
  position = "identity",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
```

```
  lineend = "round",
  show_error = TRUE,
  center_symbol = "bar",
  center_width = 0.3,
  error_width = 0.2,
  center_func = mean,
  low_func = function(x, na.rm) {
     mean(x, na.rm = na.rm) - sd(x, na.rm = na.rm)
 },
  high_func = function(x, na.rm) {
     mean(x, na.rm = na.rm) + sd(x, na.rm = na.rm)
 },
  ...
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer, either as a ggproto Geom subclass or as a string naming the stat stripped of the stat_ prefix (e.g. "count" rather than "stat_count")
position	Position adjustment, either as a string naming the adjustment (e.g. "jitter" to use position_jitter), or the result of a call to a position adjustment function. Use the latter if you need to change the settings of the adjustment.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes. It can also be a named logical vector to finely select the aesthetics to display.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders().
lineend	Line end style (round, butt, square).
show_error	show error symbol
center_symbol	one of point, bar

center_width	if center_symbol='bar', the width of the bar
error_width	the width of the error bar
center_func	the center function, mean as default
low_func	the low error function, mean minus sd as default
high_func	the high error function, mean plus sd as default
...	Other arguments passed on to ggplot2::point() or ggplot2::geom_segment.

Value

ggplot object

---

gradient_colors *generate gradient colors*

Description

generate gradient colors

Usage

```
gradient_colors(x, n)
```

Arguments

x	colors
n	number of colors to output

Value

gradient colors

Examples

```
gradient_colors(c("blue", "red"), 10)
```

inch2cm *trans inch to cm*

Description

trans inch to cm

Usage

```
inch2cm(x)

in2cm(x)

cm2in(x)
```

Arguments

x	inch value

Value

cm value

Examples

```
inch2cm(1)
```

inch2mm *trans inch to mm*

Description

trans inch to mm

Usage

```
inch2mm(x)

in2mm(x)
```

Arguments

x	inch value

Value

mm value

Examples

```
inch2mm(1)
```

`lpt` *trans geom line point and theme line point to the real point*

Description

trans geom line point and theme line point to the real point

Usage

```
lpt(x)
```

Arguments

`x`	line point in geom or theme

Value

real point

Examples

```
lpt(1)
```

`mini_diamond` *Minimal tibble dataset adjusted from diamond*

Description

Minimal tibble dataset adjusted from diamond

Usage

```
mini_diamond
```

Format

`mini_diamond`**:**
A data frame with 100 rows and 7 columns:
id unique id
cut, clarity 2 category variables
carat, price, x, y 4 continuous variables ...

Source

adjusted from `ggplot2`

mm2inch *trans mm to inch*

Description

trans mm to inch

Usage

```
mm2inch(x)

mm2in(x)
```

Arguments

x	mm value

Value

inch value

Examples

```
mm2inch(1)
```

mm2pt *trans mm to pt*

Description

trans mm to pt

Usage

```
mm2pt(x)
```

Arguments

x	mm value

Value

pt value

Examples

```
mm2pt(1)
```

plot_colors *plot colors*

Description

plot colors

Usage

```
plot_colors(x, ncol = 10, show_name = TRUE)
```

Arguments

x	color values
ncol	color number of each row
show_name	use vector names as label, FALSE to show the color value

Value

ggplot object

Examples

```
plot_colors(gradient_colors(c("blue", "red"), 10))
```

pl_init *set size, resolution and default theme*

Description

set size, resolution and default theme

Usage

```
pl_init(
  width = 4,
  height = 3,
  res = 300,
  w = NULL,
  h = NULL,
  theme = theme_pl()
)
```

Arguments

width	width
height	height
res	resolution, 300 as default
w	alias of width
h	alias of height
theme	default theme

Value

no return value

Examples

```
pl_init()
```

---

pl_save *save plot, support save into a blank canvas*

Description

save plot, support save into a blank canvas

Usage

```
pl_save(
  plot,
  filename,
  width,
  height,
  units = "in",
  canvas = NULL,
  canvas_pos_x = 0.5,
  canvas_pos_y = 0.1,
  ...
)
```

Arguments

plot	ggplot object
filename	filename
width	plot width
height	plot height
units	units, 'in' for inch as default. Can be 'in', 'cm'

canvas	NULL as default, pass character to use built-in canvas ('A4', 'A4v'), or pass a numeric vector in 'c(width, heigh)' form
canvas_pos_x	from 0 to 1, the horizontal position of plot in canvas
canvas_pos_y	from 0 to 1, the vertical position of plot in canvas
...	other arguments from ggsave

Value

no return value

`pl_size` *set repr size and resolution*

Description

set repr size and resolution

Usage

```
pl_size(width = 4, height = 3, res = 300, w = NULL, h = NULL)
```

Arguments

width	width
height	height
res	resolution, 300 as default
w	alias of width
h	alias of height

Value

no return value

Examples

```
pl_size(width = 3, height = 2)
```

PositionFloatxPL *PositionFloatxPL*

Description

PositionFloatxPL

Usage

```
PositionFloatxPL
```

Format

An object of class PositionFloatxPL (inherits from Position, ggproto, gg) of length 4.

PositionFloatyPL *PositionFloatyPL*

Description

PositionFloatyPL

Usage

```
PositionFloatyPL
```

Format

An object of class PositionFloatyPL (inherits from Position, ggproto, gg) of length 4.

position_floatxPL *a new Position object to create float x position*

Description

a new Position object to create float x position

Usage

```
position_floatxPL(float = -0.05, cycle = 2)
```

Arguments

float	float range, a ratio according to the whole panel height
cycle	float cycle

Value

Position object

`position_floatyPL` *a new Position object to create float y position*

Description

a new Position object to create float y position

Usage

```
position_floatyPL(float = -0.05, cycle = 2)
```

Arguments

`float`	float range, a ratio according to the whole panel height
`cycle`	float cycle

Value

Position object

`pt2cm` *trans pt to cm*

Description

trans pt to cm

Usage

```
pt2cm(x)
```

Arguments

`x`	pt value

Value

cm value

Examples

```
pt2cm(1)
```

pt2mm *trans pt to mm*

Description

trans pt to mm

Usage

```
pt2mm(x)
```

Arguments

x	pt value

Value

mm value

Examples

```
pt2mm(1)
```

revert_pos_scale *revert the position scale transformation*

Description

revert the position scale transformation

Usage

```
revert_pos_scale(s)
```

Arguments

s	ScaleContinuousPosition object, e.g. scales$y in compute_group()

Value

function

scale_ele *scale element according to a vector of element scales*

Description

scale element according to a vector of element scales

Usage

```
scale_ele(level, base, ele_scales)
```

Arguments

level	output level
base	value of base level
ele_scales	vector of element scales

Value

value of output level

Examples

```
scale_ele(level = 2, base = 5, ele_scales = c(1, 2))
```

scale_x_continuous_pl *A variant of* scale_x_continuous() *to show axis minor breaks*

Description

A variant of scale_x_continuous() to show axis minor breaks

Usage

```
scale_x_continuous_pl(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = NULL,
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = ggplot2::expansion(),
  oob = scales::oob_keep,
  na.value = NA_real_,
  trans = "identity",
```

```
  guide = ggh4x::guide_axis_minor(),
  position = "bottom",
  sec.axis = waiver(),
  show_minor_breaks = TRUE,
  minor_break_step = NULL
)
```

Arguments

name	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
breaks	One of: • `NULL` for no breaks • `waiver()` for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.
minor_breaks	One of: • `NULL` for no minor breaks • `waiver()` for the default breaks (one minor break between each major break) • A numeric vector of positions • A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation.
n.breaks	An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.
labels	One of: • `NULL` for no labels • `waiver()` for the default labels computed by the transformation object • A character vector giving labels (must be same length as `breaks`) • An expression vector (must be the same length as breaks). See ?plotmath for details. • A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.
limits	One of: • `NULL` to use the default scale range • A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum

- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`expand` use `expansion()` to dismiss the blank between x axis low limit and y axis

`oob` use `scales::oob_keep` instead of `scales::oob_censor`, which will always consider the data points out of the limits

`na.value` Missing values will be replaced with this value.

`trans` For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called `<name>_trans` (e.g., `scales::boxcox_trans()`). You can create your own transformation with `scales::trans_new()`.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`position` For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

`sec.axis` `sec_axis()` is used to specify a secondary axis.

`show_minor_breaks`
show minor breaks or not

`minor_break_step`
the step of minor breaks

Value

scale object

`scale_x_log10_pl` *A variant of* `scale_x_log10()` *to show axis minor breaks and better axis labels*

Description

A variant of `scale_x_log10()` to show axis minor breaks and better axis labels

Usage

```
scale_x_log10_pl(
  name = waiver(),
  breaks = NULL,
  minor_breaks = NULL,
  n.breaks = NULL,
  labels = NULL,
  limits = NULL,
  expand = ggplot2::expansion(),
  oob = scales::oob_keep,
  na.value = NA_real_,
  trans = scales::log10_trans(),
  guide = ggh4x::guide_axis_minor(),
  position = "bottom",
  sec.axis = waiver(),
  show_minor_breaks = TRUE
)
```

Arguments

name	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
breaks	One of: • `NULL` for no breaks • `waiver()` for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.
minor_breaks	One of: • `NULL` for no minor breaks • `waiver()` for the default breaks (one minor break between each major break) • A numeric vector of positions • A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation.
n.breaks	An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.
labels	One of: • `NULL` for no labels • `waiver()` for the default labels computed by the transformation object • A character vector giving labels (must be same length as `breaks`)

- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale range
- A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`expand` use `expansion()` to dismiss the blank between x axis low limit and y axis

`oob` use `scales::oob_keep` instead of `scales::oob_censor`, which will always consider the data points out of the limits

`na.value` Missing values will be replaced with this value.

`trans` For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called `<name>_trans` (e.g., `scales::boxcox_trans()`). You can create your own transformation with `scales::trans_new()`.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`position` For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

`sec.axis` `sec_axis()` is used to specify a secondary axis.

`show_minor_breaks` show minor breaks or not

Value

scale object

`scale_y_continuous_pl` *A variant of* `scale_y_continuous()` *to show axis minor breaks*

Description

A variant of `scale_y_continuous()` to show axis minor breaks

Usage

```
scale_y_continuous_pl(
  name = waiver(),
  breaks = waiver(),
  minor_breaks = NULL,
  n.breaks = NULL,
  labels = waiver(),
  limits = NULL,
  expand = ggplot2::expansion(),
  oob = scales::oob_keep,
  na.value = NA_real_,
  trans = "identity",
  guide = ggh4x::guide_axis_minor(),
  position = "left",
  sec.axis = waiver(),
  show_minor_breaks = TRUE,
  minor_break_step = NULL
)
```

Arguments

name	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
breaks	One of: • `NULL` for no breaks • `waiver()` for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.
minor_breaks	One of: • `NULL` for no minor breaks • `waiver()` for the default breaks (one minor break between each major break) • A numeric vector of positions • A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation.
n.breaks	An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.
labels	One of: • `NULL` for no labels • `waiver()` for the default labels computed by the transformation object

- A character vector giving labels (must be same length as `breaks`)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale range
- A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`expand` use `expansion()` to dismiss the blank between y axis low limit and x axis

`oob` use `scales::oob_keep` instead of `scales::oob_censor`, which will always consider the data points out of the limits

`na.value` Missing values will be replaced with this value.

`trans` For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called `<name>_trans` (e.g., `scales::boxcox_trans()`). You can create your own transformation with `scales::trans_new()`.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`position` For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

`sec.axis` `sec_axis()` is used to specify a secondary axis.

`show_minor_breaks` show minor breaks or not

`minor_break_step` the step of minor breaks

Value

scale object

`scale_y_log10_pl` *A variant of* `scale_y_log10()` *to show axis minor breaks and better axis labels*

Description

A variant of `scale_y_log10()` to show axis minor breaks and better axis labels

Usage

```
scale_y_log10_pl(
  name = waiver(),
  breaks = NULL,
  minor_breaks = NULL,
  n.breaks = NULL,
  labels = NULL,
  limits = NULL,
  expand = ggplot2::expansion(),
  oob = scales::oob_keep,
  na.value = NA_real_,
  trans = scales::log10_trans(),
  guide = ggh4x::guide_axis_minor(),
  position = "left",
  sec.axis = waiver(),
  show_minor_breaks = TRUE
)
```

Arguments

`name`	The name of the scale. Used as the axis or legend title. If `waiver()`, the default, the name of the scale is taken from the first mapping used for that aesthetic. If `NULL`, the legend title will be omitted.
`breaks`	One of: • `NULL` for no breaks • `waiver()` for the default breaks computed by the transformation object • A numeric vector of positions • A function that takes the limits as input and returns breaks as output (e.g., a function returned by `scales::extended_breaks()`). Also accepts rlang lambda function notation.
`minor_breaks`	One of: • `NULL` for no minor breaks • `waiver()` for the default breaks (one minor break between each major break) • A numeric vector of positions

- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation.

`n.breaks` An integer guiding the number of major breaks. The algorithm may choose a slightly different number to ensure nice break labels. Will only have an effect if `breaks = waiver()`. Use `NULL` to use the default number of breaks given by the transformation.

`labels` One of:

- `NULL` for no labels
- `waiver()` for the default labels computed by the transformation object
- A character vector giving labels (must be same length as `breaks`)
- An expression vector (must be the same length as breaks). See ?plotmath for details.
- A function that takes the breaks as input and returns labels as output. Also accepts rlang lambda function notation.

`limits` One of:

- `NULL` to use the default scale range
- A numeric vector of length two providing limits of the scale. Use `NA` to refer to the existing minimum or maximum
- A function that accepts the existing (automatic) limits and returns new limits. Also accepts rlang lambda function notation. Note that setting limits on positional scales will **remove** data outside of the limits. If the purpose is to zoom, use the limit argument in the coordinate system (see `coord_cartesian()`).

`expand` use `expansion()` to dismiss the blank between y axis low limit and x axis

`oob` use `scales::oob_keep` instead of `scales::oob_censor`, which will always consider the data points out of the limits

`na.value` Missing values will be replaced with this value.

`trans` For continuous scales, the name of a transformation object or the object itself. Built-in transformations include "asn", "atanh", "boxcox", "date", "exp", "hms", "identity", "log", "log10", "log1p", "log2", "logit", "modulus", "probability", "probit", "pseudo_log", "reciprocal", "reverse", "sqrt" and "time".

A transformation object bundles together a transform, its inverse, and methods for generating breaks and labels. Transformation objects are defined in the scales package, and are called `<name>_trans` (e.g., `scales::boxcox_trans()`). You can create your own transformation with `scales::trans_new()`.

`guide` A function used to create a guide or its name. See `guides()` for more information.

`position` For position scales, The position of the axis. `left` or `right` for y axes, `top` or `bottom` for x axes.

`sec.axis` `sec_axis()` is used to specify a secondary axis.

`show_minor_breaks`
show minor breaks or not

Value

scale object

---

sci_colors *select colors from* ggsci *package presets*

Description

select colors from ggsci package presets

Usage

```
sci_colors(name, n = 3, alpha = 1)
```

Arguments

name	presets name
n	number of colors
alpha	alpha

Value

colors

Examples

```
sci_colors("npg", 5)
```

StatCompare *StatCompare*

Description

StatCompare

Usage

```
StatCompare
```

Format

An object of class StatCompare (inherits from Stat, ggproto, gg) of length 6.

StatCountPL *StatCountPL*

Description

StatCountPL

Usage

```
StatCountPL
```

Format

An object of class StatCountPL (inherits from Stat, ggproto, gg) of length 4.

StatDescribe *StatDescribe*

Description

StatDescribe

Usage

```
StatDescribe
```

Format

An object of class StatDescribe (inherits from Stat, ggproto, gg) of length 4.

StatFuncPL *StatFuncPL*

Description

StatFuncPL

Usage

```
StatFuncPL
```

Format

An object of class StatFuncPL (inherits from Stat, ggproto, gg) of length 4.

StatMeanPL *StatMeanPL*

Description

StatMeanPL

Usage

```
StatMeanPL
```

Format

An object of class `StatMeanPL` (inherits from `Stat`, `ggproto`, `gg`) of length 4.

theme_pl *a new extensible theme*

Description

a new extensible theme

Usage

```
theme_pl(
  base_size = 10,
  base_line_size = lpt(base_size/10),
  base_rect_size = lpt(1),
  size_scales = c(5, 6, 7),
  margin_factor = 0.25,
  plot_margin_factor = 0.5,
  legend_spacing_factor = 1.2,
  font_family = "",
  ...
)
```

Arguments

`base_size`	base size of fonts and margins
`base_line_size`	base linewidth
`base_rect_size`	base linewidth of the rectangles
`size_scales`	a vector of element size scales, namely: 1. base size, used by legend text, axis text, caption 2. used by legend title, axis title, strip text (facet title), subtitle

3. used by title, tag

`margin_factor` factor to adjust the element margins according to size_scales

`plot_margin_factor` factor to adjust the plot margins according to `size_scales[3]`

`legend_spacing_factor` factor to adjust the space of legend items according to `size_scales[2]`

`font_family` font family

`...` arguments from ggplot2::theme()

Value

theme object of ggplot

Examples

```
theme_pl()
```

---

theme_pl0 *a blank theme*

Description

a blank theme

Usage

```
theme_pl0(...)
```

Arguments

`...` arguments of theme_pl()

Value

theme object of ggplot

Examples

```
theme_pl0()
```

tpt *trans geom text or point to the real point*

Description

trans geom text or point to the real point

Usage

```
tpt(x)

ppt(x)
```

Arguments

x	text point in geom

Value

real point

Examples

```
tpt(1)
```

trans_pos_scale *perform the position scale transformation*

Description

perform the position scale transformation

Usage

```
trans_pos_scale(s)
```

Arguments

s	ScaleContinuousPosition object, e.g. scales$y in compute_group()

Value

function

Index