Package 'ggprism'

March 21, 2024

Title A 'ggplot2' Extension Inspired by 'GraphPad Prism'

Version 1.0.5

Description Provides various themes, palettes, and other functions that are used to customise ggplots to look like they were made in 'GraphPad Prism'. The 'Prism'-look is achieved with theme_prism() and scale_fill|colour_prism(), axes can be changed with custom guides like guide_prism_minor(), and significance indicators added with add_pvalue().

License GPL (>= 3.0)

URL https://csdaw.github.io/ggprism/, https://github.com/csdaw/ggprism

BugReports https://github.com/csdaw/ggprism/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

Depends R (>= 3.2)

Imports digest, ggplot2 (>= 3.2.0), glue, grid, gtable (>= 0.1.1), rlang (>= 0.3.0), scales (>= 0.5.0), stats, tibble, utils

Suggests covr, dplyr, ggbeeswarm, ggnewscale, knitr, magrittr, patchwork, rmarkdown, rstatix, tidyr, tinytest

VignetteBuilder knitr

NeedsCompilation no

Author Charlotte Dawson [aut, cre] (<https://orcid.org/0000-0002-7151-5971>)

Maintainer Charlotte Dawson <csdaw@outlook.com>

Repository CRAN

Date/Publication 2024-03-21 10:50:02 UTC

R topics documented:

add_pvalue 2
annotation_ticks 9

ggprism_data 12
guide_prism_bracket 12
guide_prism_minor 14
guide_prism_offset 17
guide_prism_offset_minor 19
preview_theme 22
prism_colour_pal 23
prism_fill_pal 24
prism_shape_pal 25
scale_colour_prism 26
scale_fill_prism 28
scale_shape_prism 31
theme_prism 34
wings 36

Index **38**

`add_pvalue` *Add p-values to a ggplot*

Description

Add p-values with or without brackets to a ggplot.

See here or the examples section below for examples of how to use.

`add_pvalue` is a refactored version of `stat_pvalue_manual` from kassambara/ggpubr, altered to have less dependencies, and more flexibility with input format and aesthetics. Any examples using `stat_pvalue_manual` found on Datanovia will also work with `add_pvalue`.

Usage

```
add_pvalue(
  data,
  label = NULL,
  xmin = "group1",
  xmax = "group2",
  x = NULL,
  y.position = "y.position",
  parse = FALSE,
  label.size = 3.2,
  colour = NULL,
  color = NULL,
  tip.length = 0.03,
  bracket.size = 0.6,
  bracket.colour = NULL,
  bracket.color = NULL,
  bracket.shorten = 0,
  bracket.nudge.y = 0,
```

```
  step.increase = 0,
  step.group.by = NULL,
  remove.bracket = FALSE,
  coord.flip = FALSE,
  position = "identity",
  ...
)
```

Arguments

data	A data.frame with the statistics to plot. Default format has the following columns: group1 | group2 | p.adj | y.position | etc. group1 and group2 are the two groups that were compared. p.adj is the adjusted p-value. y.position is the y coordinate that specifies where on the plot the p-value should go. The column names can differ from the default as long as their are specified when calling the function.
label	string. Name of column in data that contains the text to plot (e.g. label = "p.adj"). Can also be an expression that can be formatted by glue (e.g. label = "p = {p.adj}").
xmin	string. Name of column in data that contains the position of the left side of the brackets. Default is "group1".
xmax	Optional. string. Name of column in data that contains the position of the right side of the brackets. Default is "group2". If NULL, the p-values are plotted as text only.
x	string or numeric. x coordinate of the p-value text. Only use when plotting p-value as text only (no brackets).
y.position	string. Name of column in data containing the y coordinates (numeric) of each p-value to be plotted. Can also be a single number to plot all p-values at the same height or a numeric vector that will override data.
parse	logical. Default is FALSE. If TRUE the text labels will be parsed into expressions and displayed as described in ?plotmath.
label.size	numeric. Size of text.
colour, color	string. Colour of text.
tip.length	numeric vector. Length of bracket tips. Use 0 to remove tips.
bracket.size	numeric. Line width of bracket.
bracket.colour, bracket.color	string. Colour of bracket. Default is NULL which causes brackets to inherit the colour of the text.
bracket.shorten	numeric. Shortens the brackets slightly to allow them to be plotted side-by-side at the same y position.
bracket.nudge.y	numeric. Changes the y position of p-values. Useful for slightly adjusting p-values if the text is cut off.
step.increase	numeric. Changes the space between brackets.

step.group.by	string. Variable to group brackets by.
remove.bracket	logical. If TRUE all brackets are removed and p-value is shown as text only.
coord.flip	logical. If TRUE p-values are rotated by 90 degrees. Should be used with coord_flip
position	string or call to position function such as position_dodge. Typically used for adjusting x position of p-values to be in line with dodged data.
...	Additional aesthetics or arguments passed to layer. See below for allowed values.

Value

Returns a *layer* ggproto object with either geom = GeomBracket or geom = GeomText.

Allowed ... values

add_pvalue understands the following additional aesthetics or arguments:

fontface string. Fontface of text (e.g. "bold").

fontfamily string. Fontfamily of text (e.g. "Arial").

hjust numeric. Horizontal justification of text.

vjust numeric. Vertical justification of text.

alpha numeric. Transparency of text and/or brackets.

linetype string or numeric. Linetype of brackets (e.g. "dashed").

lineend string. Lineend of brackets (e.g. "butt").

na.rm logical. If FALSE (default), removes missing values with a warning. If TRUE silently removes missing values.

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

inherit.aes logical. If FALSE, overrides the default aesthetics, rather than combining with them.

Examples

```
library(ggplot2)

## we will use the ToothGrowth dataset for all examples
tg <- ToothGrowth
tg$dose <- as.factor(tg$dose)
tg$group <- factor(rep(c("grp1", "grp2"), 30))

## p-value bracket comparing two means
# p-value table (its best to use these column names)
two.means <- tibble::tribble(
  ~group1, ~group2, ~p,     ~y.position,
  "OJ",    "VC",    0.0606, 36
```

```
)

# boxplot (or another geom...)
ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  add_pvalue(two.means)

# if your table has special column names you will need to specify them
two.means <- tibble::tribble(
  ~apple, ~banana, ~my.pval, ~some.y.position,
  "OJ",    "VC",    0.0606,  36
)

ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  add_pvalue(
    two.means,
    xmin = "apple",
    xmax = "banana",
    label = "my.pval",
    y.position = "some.y.position"
  )

## you can make the label a glue expression
two.means <- tibble::tribble(
  ~group1, ~group2, ~p,     ~y.position,
  "OJ",    "VC",    0.0606, 36
)

ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  add_pvalue(two.means, label = "p = {p}")

## you can change aesthetics of the bracket and label
ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  add_pvalue(
    two.means,
    label = "p = {p}",
    colour = "red", # label
    label.size = 6, # label
    fontface = "bold", # label
    fontfamily = "serif", # label
    angle = 45, # label
    bracket.colour = "blue", # bracket
    bracket.size = 1, # bracket
    linetype = "dashed", # bracket
    lineend = "round" # bracket
  )

## you can change the tip length of the bracket
# make them longer
ggplot(tg, aes(x = supp, y = len)) +
```

```
  geom_boxplot() +
  add_pvalue(two.means, tip.length = 0.1)

# make them disappear
ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  add_pvalue(two.means, tip.length = 0)

# make one side longer than the other
ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  add_pvalue(two.means, tip.length = c(0.1, 0))

## p-value brackets with comparisons to a reference sample
each.vs.ref <- tibble::tribble(
  ~group1, ~group2, ~p.adj,   ~y.position,
  "0.5",   "1",     8.80e-14, 35,
  "0.5",   "2",     1.27e-7,  38
)

ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = dose)) +
  add_pvalue(each.vs.ref)

## p-value brackets with pairwise comparisons
pairwise <- tibble::tribble(
  ~group1, ~group2,  ~p.signif, ~y.position,
  "0.5",   "1",      "****",    38,
  "0.5",   "2",      "****",    36,
  "1",     "2",      "****",    38
)

# you can shorten the length of brackets that are close together
ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = dose)) +
  add_pvalue(
    pairwise,
    bracket.shorten = c(0.05, 0, 0.05)
  )

# you can nudge brackets that are not quite in the correct y position
# instead of changing the p-value table
ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = dose)) +
  add_pvalue(
    pairwise,
    bracket.shorten = c(0.05, 0, 0.05),
    bracket.nudge.y = c(0.5, 0, 0.5)
  )

## p-value brackets with pairwise comparisons of grouped data
pairwise.grouped <- tibble::tribble(
  ~group1, ~group2, ~p.adj,  ~y.position, ~supp,
```

```
  "0.5",   "1",     2.63e-4, 33.5,        "OJ",
  "0.5",   "2",     3.96e-6, 37.6,        "OJ",
  "1",     "2",     1.18e-1, 41.6,        "OJ",
  "0.5",   "1",     2.04e-6, 36.5,        "VC",
  "0.5",   "2",     1.40e-7, 40.6,        "VC",
  "1",     "2",     2.75e-4, 44.6,        "VC"
)

# use step.increase to change the spacing between different brackets in the
# groups specified by step.group.by
ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = supp)) +
  add_pvalue(
    pairwise.grouped,
    colour = "supp",
    tip.length = 0,
    step.group.by = "supp",
    step.increase = 0.03
  )

## p-value (brackets) with single facet variable
two.means.grouped1 <- tibble::tribble(
  ~group1, ~group2, ~p.adj,  ~y.position, ~dose,
  "OJ",    "VC",    0.0127,  24,          "0.5",
  "OJ",    "VC",    0.00312, 30,          "1",
  "OJ",    "VC",    0.964,   36.5,        "2"
)

ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  facet_wrap(~ dose, scales = "free") +
  add_pvalue(two.means.grouped1) # table must have dose column

## p-value (brackets) with single facet variable and multiple brackets per facet
pairwise.grouped <- tibble::tribble(
  ~group1, ~group2, ~p.adj,  ~y.position, ~supp,
  "0.5",   "1",     2.63e-4, 33.5,        "OJ",
  "0.5",   "2",     3.96e-6, 37.6,        "OJ",
  "1",     "2",     1.18e-1, 41.6,        "OJ",
  "0.5",   "1",     2.04e-6, 36.5,        "VC",
  "0.5",   "2",     1.40e-7, 40.6,        "VC",
  "1",     "2",     2.75e-4, 44.6,        "VC"
)

ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = supp)) +
  facet_wrap(~ supp) +
  add_pvalue(pairwise.grouped)

## p-value (brackets) with two facet variables
two.means.grouped2 <- tibble::tribble(
  ~group1, ~group2, ~p.signif, ~y.position, ~group, ~dose,
  "OJ",    "VC",    "*",       21,          "grp1", "0.5",
```

```
  "OJ",    "VC",    "**",      30,          "grp2", "1"
)

ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  facet_wrap(group ~ dose) +
  add_pvalue(two.means.grouped2) # table must have dose and group column

## p-value (text only) comparing two means
two.means <- tibble::tribble(
  ~group1, ~group2, ~p,     ~y.position,
  "OJ",    "VC",    0.0606, 36
)

ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  add_pvalue(two.means, remove.bracket = TRUE, x = 1.5)

## p-value (text only) with coord_flip, override y.position, change angle
ggplot(tg, aes(x = supp, y = len)) +
  geom_boxplot() +
  add_pvalue(
    two.means,
    remove.bracket = TRUE,
    x = 1.5,
    y.position = 32,
    angle = 45
  ) +
  coord_flip()

## p-value (text only) comparing to the null
one.mean <- tibble::tribble(
  ~group1, ~group2,      ~p.signif, ~y.position, ~dose,
  "1",     "null model", "****",    35,          "0.5",
  "1",     "null model", "****",    35,          "1",
  "1",     "null model", "****",    35,          "2"
)

ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = dose)) +
  add_pvalue(one.mean, x = "dose")

## p-value (text only) with comparisons to a base mean
each.vs.basemean <- tibble::tribble(
  ~group1, ~group2, ~p.adj, ~y.position,
  "all",   "0.5",   "****", 35,
  "all",   "1",     "ns",   35,
  "all",   "2",     "****", 35
)

ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = dose)) +
  add_pvalue(each.vs.basemean)
```

```
## p-value (text only) with comparison to reference sample
each.vs.ref <- tibble::tribble(
  ~group1, ~group2, ~p.adj,   ~y.position,
  "0.5",   "1",     8.80e-14, 35,
  "0.5",   "2",     1.27e-7,  38
)

ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = dose)) +
  add_pvalue(each.vs.ref, coord.flip = TRUE, remove.bracket = TRUE)

## p-value (text only) with a grouping variable
two.means.grouped1 <- tibble::tribble(
  ~group1, ~group2, ~p.adj,  ~y.position, ~dose,
  "OJ",    "VC",    0.0127,  24,          "0.5",
  "OJ",    "VC",    0.00312, 30,          "1",
  "OJ",    "VC",    0.964,   36.5,        "2"
)

ggplot(tg, aes(x = dose, y = len)) +
  geom_boxplot(aes(fill = supp)) +
  add_pvalue(two.means.grouped1, x = "dose")
```

`annotation_ticks` *Add ticks as ggplot annotation*

Description

This is an annotation function to add tick marks (major, minor, or both) to a ggplot. Clipping must be turned off if the ticks are to appear outside the plotting area, for example with: `coord_cartesian(clip = "off")`.

Usage

```
annotation_ticks(
  sides = "b",
  type = "both",
  outside = FALSE,
  tick.length = unit(4.8, "pt"),
  minor.length = unit(2.4, "pt"),
  linewidth = 0.6,
  colour = "black",
  color = NULL,
  linetype = 1,
  lineend = "butt",
  alpha = 1,
  data = data.frame(x = NA)
)
```

Arguments

sides	string. Indicates which sides of the plot should ticks appear. Can be any of "trbl", for top, right, bottom, left.
type	string. Types of ticks that appear. One of "major", "minor", or "both". Control number of ticks by controlling the breaks and minor_breaks arguments in the various ggplot2 scale_(x|y)_ functions.
outside	logical. Should the ticks point outside of the plotting area? If TRUE clipping must be turned off.
tick.length	a unit object specifying the length of major ticks.
minor.length	a unit object specifying the length of minor ticks.
linewidth	numeric. Linewidth of ticks.
colour, color	string. Colour of ticks.
linetype	string or numeric. Linetype of tick marks.
lineend	string. Lineend of ticks. One of "square" (default), "butt", or "round".
alpha	numeric. Transparency of ticks.
data	data.frame. Use this argument to control the appearance of ticks on different facets. Pass a data.frame containing the levels from the faceting variable you want to annotate specifically. See here for an example.

Value

Returns a *layer* ggproto object with geom = GeomTicks.

Source

The code is a slightly modified version of the answer to this Stack Overflow question, which is itself a refactored version of this annotation_ticks() function.

Examples

```
## Generally it is better to use the guide_prism_minor function.
## However annotation_ticks is useful in a few specific situations.
library(ggplot2)

## easily put ticks without labels around a plot with a border
ggplot(mtcars, aes(x = mpg, y = disp)) +
  geom_point() +
  theme_prism(border = TRUE) +
  coord_cartesian(clip = "off") +
  annotation_ticks(sides = "tr", type = "major", outside = TRUE) +
  theme(plot.margin = unit(c(4, 4, 4, 4), "mm"))

# the same but with minor ticks as well
ggplot(mtcars, aes(x = mpg, y = disp)) +
  geom_point() +
  scale_x_continuous(guide = "prism_minor") +
  scale_y_continuous(guide = "prism_minor") +
```

```
  theme_prism(border = TRUE) +
  coord_cartesian(clip = "off") +
  annotation_ticks(sides = "tr", type = "both", outside = TRUE) +
  theme(plot.margin = unit(c(4, 4, 4, 4), "mm"))

# you can adjust the appearance of annotation_ticks
ggplot(mtcars, aes(x = mpg, y = disp)) +
  geom_point() +
  theme_prism(border = TRUE) +
  coord_cartesian(clip = "off") +
  annotation_ticks(
    sides = "tr",
    type = "major",
    outside = TRUE,
    tick.length = unit(10, "pt"),
    colour = "red",
    linewidth = 2,
    linetype = "dashed",
    lineend = "round"
  ) +
  theme(plot.margin = unit(c(4, 4, 4, 4), "mm"))

## Unfortunately, due to the way they work, secondary axes don't always play
## well with the minor tick axes guides in this package.
## So we can use annotation_ticks instead.
sample.data <- data.frame(
  day = as.Date("2019-01-01") + 0:99,
  temperature = runif(100) + seq(1, 100)^2.5 / 10000,
  price = runif(100) + seq(100, 1)^1.5 / 10
)

# sample graph with secondary axis
ggplot(sample.data, aes(x = day)) +
  geom_line(aes(y = temperature), colour = "magenta") +
  geom_line(aes(y = price / 10), colour = "blue") +
  scale_y_continuous(sec.axis = sec_axis(~. * 10, name = "price")) +
  theme_prism(border = TRUE) +
  coord_cartesian(clip = "off")

# guide_prism_minor only works with the main axis in this case
ggplot(sample.data, aes(x = day)) +
  geom_line(aes(y = temperature), colour = "magenta") +
  geom_line(aes(y = price / 10), colour = "blue") +
  scale_y_continuous(
    sec.axis = sec_axis(~. * 10, name = "price"),
    guide = "prism_minor"
  ) +
  theme_prism(border = TRUE) +
  coord_cartesian(clip = "off")

# we use annotation_ticks to draw the minor ticks on the secondary axis
ggplot(sample.data, aes(x = day)) +
  geom_line(aes(y = temperature), colour = "magenta") +
```

```
  geom_line(aes(y = price / 10), colour = "blue") +
  scale_y_continuous(
    sec.axis = sec_axis(~. * 10, name = "price"),
    guide = "prism_minor"
  ) +
  theme_prism(border = TRUE) +
  coord_cartesian(clip = "off") +
  annotation_ticks(sides = "r", type = "minor", outside = TRUE)
```

---

ggprism_data *Palettes and theme data for ggprism*

Description

This list object contains the strings and values used in ggprism themes and palettes.

Usage

```
ggprism_data
```

Format

An object of class `list` of length 4.

---

guide_prism_bracket *Axis guide with brackets*

Description

This guide turns the axis into brackets drawn around each axis label.

Usage

```
guide_prism_bracket(
  title = waiver(),
  check.overlap = FALSE,
  angle = NULL,
  n.dodge = 1,
  order = 0,
  position = waiver(),
  width = NULL,
  outside = TRUE
)
```

Arguments

title	A character string or expression indicating a title of guide. If NULL, the title is not shown. By default (waiver()), the name of the scale object or the name specified in labs() is used for the title.
check.overlap	silently remove overlapping labels, (recursively) prioritizing the first, last, and middle labels.
angle	Compared to setting the angle in theme() / element_text(), this also uses some heuristics to automatically pick the hjust and vjust that you probably want. Can be one of the following: • NULL to take the angles and hjust/vjust directly from the theme. • waiver() to allow reasonable defaults in special cases. • A number representing the text angle in degrees.
n.dodge	The number of rows (for vertical axes) or columns (for horizontal axes) that should be used to render the labels. This is useful for displaying labels that would otherwise overlap.
order	A positive integer of length 1 that specifies the order of this guide among multiple guides. This controls in which order guides are merged if there are multiple guides for the same position. If 0 (default), the order is determined by a secret algorithm.
position	Where this guide should be drawn: one of top, bottom, left, or right.
width	numeric. Controls the width of the bracket. Try values between 0 and 1.
outside	logical. Default is TRUE and brackets point outwards. If FALSE the bracket crossbar is moved so the ticks appear to point inwards towards the plotting area.

Details

The number of brackets can be adjusted using the breaks argument in scale_(x|y)_continuous() or scale_(x|y)_discrete().

Value

Returns a *prism_bracket* guide class object.

Examples

```
library(ggplot2)

## base plot
base <- ggplot(mpg, aes(x = as.factor(cyl), y = hwy)) +
  geom_jitter(width = 0.2) +
  theme(axis.line = element_line(colour = "black"))

## use brackets on x axis
# if not specified, the width of the brackets is guessed
base + scale_x_discrete(guide = "prism_bracket")

# you can add brackets using the guide function as well
```

```
base + guides(x = "prism_bracket")

## works with coord_flip
base + scale_x_discrete(guide = "prism_bracket") +
  coord_flip()

## adjust bracket width
base + scale_x_discrete(guide = guide_prism_bracket(width = 0.12))

## make brackets point inward
base + scale_x_discrete(guide = guide_prism_bracket(width = 0.12, outside = FALSE))

## change colour with the usual axis.line, axis.ticks, axis.text elements
base + scale_x_discrete(guide = guide_prism_bracket(width = 0.12, outside = FALSE)) +
  theme(axis.line.x = element_line(colour = "red"),
        axis.ticks.x = element_line(colour = "blue"),
        axis.text.x = element_text(colour = "green"))
```

`guide_prism_minor` *Axis guide with minor ticks*

Description

This guide is like the standard `guide_axis`, but with minor ticks.

Usage

```
guide_prism_minor(
  title = waiver(),
  check.overlap = FALSE,
  angle = NULL,
  n.dodge = 1,
  order = 0,
  position = waiver()
)
```

Arguments

`title`	A character string or expression indicating a title of guide. If `NULL`, the title is not shown. By default (`waiver()`), the name of the scale object or the name specified in `labs()` is used for the title.
`check.overlap`	silently remove overlapping labels, (recursively) prioritizing the first, last, and middle labels.
`angle`	Compared to setting the angle in `theme()` / `element_text()`, this also uses some heuristics to automatically pick the `hjust` and `vjust` that you probably want. Can be one of the following: • `NULL` to take the angles and `hjust`/`vjust` directly from the theme.

- `waiver()` to allow reasonable defaults in special cases.
- A number representing the text angle in degrees.

`n.dodge` The number of rows (for vertical axes) or columns (for horizontal axes) that should be used to render the labels. This is useful for displaying labels that would otherwise overlap.

`order` A positive `integer` of length 1 that specifies the order of this guide among multiple guides. This controls in which order guides are merged if there are multiple guides for the same position. If 0 (default), the order is determined by a secret algorithm.

`position` Where this guide should be drawn: one of top, bottom, left, or right.

Details

The number of minor ticks can be changed using the `minor_breaks` argument. Control the length of minor ticks by setting `prism.ticks.length` to a `unit` object using `theme`, for example: `prism.ticks.length = unit(2, "pt")`. The major tick lengths are adjusted using the standard `axis.ticks.length`.

Value

Returns a *prism_minor* guide class object.

Examples

```
library(ggplot2)

## base plot
base <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point()

## add minor ticks to x and y axes
base +
  scale_x_continuous(
    limits = c(0, 6),
    guide = "prism_minor"
  ) +
  scale_y_continuous(
    limits = c(10, 35),
    guide = "prism_minor"
  )

## you can also use the guides function to add minor ticks
base +
  guides(x = "prism_minor", y = "prism_minor")

## adjust number of minor ticks by adjusting minor breaks
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_minor"
```

```
  ) +
  scale_y_continuous(
    limits = c(10, 35),
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_minor"
  )

## adjust the length of major ticks with the usual axis.ticks.length element
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_minor"
  ) +
  scale_y_continuous(
    limits = c(10, 35),
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_minor"
  ) +
  theme(
    axis.ticks.length = unit(10, "pt")
  )

## adjust the length of minor ticks with a new prism.ticks.length element
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_minor"
  ) +
  scale_y_continuous(
    limits = c(10, 35),
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_minor"
  ) +
  theme(
    axis.ticks.length = unit(10, "pt"),
    prism.ticks.length = unit(5, "pt")
  )

## to get log10 minor ticks just use a log10 scale and set the minor breaks
ggplot(msleep, aes(bodywt, brainwt)) +
  geom_point(na.rm = TRUE) +
  scale_x_log10(limits = c(1e0, 1e4),
                minor_breaks = rep(1:9, 4)*(10^rep(0:3, each = 9)),
                guide = "prism_minor")

## change colour with the usual axis.ticks element
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_minor"
```

```
  ) +
  scale_y_continuous(
    limits = c(10, 35),
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_minor"
  ) +
  theme(
    axis.ticks.length = unit(10, "pt"),
    prism.ticks.length = unit(5, "pt"),
    axis.ticks = element_line(colour = "red")
  )
```

`guide_prism_offset` *Offset axis guide*

Description

This guide draws the axis only as wide as the outermost tick marks, similar to offset axes from Prism.

Usage

```
guide_prism_offset(
  title = waiver(),
  check.overlap = FALSE,
  angle = NULL,
  n.dodge = 1,
  order = 0,
  position = waiver()
)
```

Arguments

`title`	A character string or expression indicating a title of guide. If `NULL`, the title is not shown. By default (`waiver()`), the name of the scale object or the name specified in `labs()` is used for the title.
`check.overlap`	silently remove overlapping labels, (recursively) prioritizing the first, last, and middle labels.
`angle`	Compared to setting the angle in `theme()` / `element_text()`, this also uses some heuristics to automatically pick the `hjust` and `vjust` that you probably want. Can be one of the following: • `NULL` to take the angles and `hjust`/`vjust` directly from the theme. • `waiver()` to allow reasonable defaults in special cases. • A number representing the text angle in degrees.
`n.dodge`	The number of rows (for vertical axes) or columns (for horizontal axes) that should be used to render the labels. This is useful for displaying labels that would otherwise overlap.

order	A positive integer of length 1 that specifies the order of this guide among multiple guides. This controls in which order guides are merged if there are multiple guides for the same position. If 0 (default), the order is determined by a secret algorithm.
position	Where this guide should be drawn: one of top, bottom, left, or right.

Details

Control the length of the axis by adjusting the breaks argument in scale_(x|y)_continuous() or scale_(x|y)_discrete().

Value

Returns a *prism_offset* guide class object.

Examples

```
library(ggplot2)

## base plot
base <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  theme(axis.line = element_line(colour = "black"))

## use offset guide via scale_x/y_continuous
base +
  scale_x_continuous(
    limits = c(1, 6),
    breaks = seq(1, 6, by = 1),
    guide = "prism_offset"
  ) +
  scale_y_continuous(
    guide = "prism_offset"
  )

## use offset guide via guides argument
base +
  guides(x = "prism_offset", y = "prism_offset") +
  scale_x_continuous(
    limits = c(1, 6),
    breaks = seq(1, 6, by = 1)
  )

## change colour and tick length with the usual elements
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_offset"
  ) +
  scale_y_continuous(
    limits = c(10, 35),
```

```
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_offset"
  ) +
  theme(
    axis.ticks.length = unit(10, "pt"),
    axis.ticks = element_line(colour = "red"),
    axis.line = element_line(colour = "blue")
  )
```

`guide_prism_offset_minor` *Offset axis guide with minor ticks*

Description

This guide draws the axis only as wide as the outermost tick marks, similar to offset axes from Prism. It also adds minor ticks.

Usage

```
guide_prism_offset_minor(
  title = waiver(),
  check.overlap = FALSE,
  angle = NULL,
  n.dodge = 1,
  order = 0,
  position = waiver()
)
```

Arguments

`title`	A character string or expression indicating a title of guide. If `NULL`, the title is not shown. By default (`waiver()`), the name of the scale object or the name specified in `labs()` is used for the title.
`check.overlap`	silently remove overlapping labels, (recursively) prioritizing the first, last, and middle labels.
`angle`	Compared to setting the angle in `theme()` / `element_text()`, this also uses some heuristics to automatically pick the `hjust` and `vjust` that you probably want. Can be one of the following: • `NULL` to take the angles and `hjust`/`vjust` directly from the theme. • `waiver()` to allow reasonable defaults in special cases. • A number representing the text angle in degrees.
`n.dodge`	The number of rows (for vertical axes) or columns (for horizontal axes) that should be used to render the labels. This is useful for displaying labels that would otherwise overlap.

order	A positive integer of length 1 that specifies the order of this guide among multiple guides. This controls in which order guides are merged if there are multiple guides for the same position. If 0 (default), the order is determined by a secret algorithm.
position	Where this guide should be drawn: one of top, bottom, left, or right.

Details

Control the length of the axis by adjusting the breaks argument in scale_(x|y)_continuous() or scale_(x|y)_discrete(). Similarly, the number of minor ticks can be changed using the minor_breaks argument.

Control the length of minor ticks by setting prism.ticks.length to a unit object using theme, for example: prism.ticks.length = unit(2, "pt"). The major tick lengths are adjusted using the standard axis.ticks.length.

Value

Returns a *prism_offset_minor* guide class object.

Examples

```
library(ggplot2)

## base plot
base <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  theme(axis.line = element_line(colour = "black"))

## add minor ticks to x and y axes
base +
  scale_x_continuous(
    limits = c(0, 6),
    guide = "prism_offset_minor"
  ) +
  scale_y_continuous(
    limits = c(10, 35),
    guide = "prism_offset_minor"
  )

## you can also use the guides function to add minor ticks
base +
  guides(x = "prism_offset_minor", y = "prism_offset_minor")

## adjust number of minor ticks by adjusting minor breaks
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_offset_minor"
  ) +
  scale_y_continuous(
```

```
    limits = c(10, 35),
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_offset_minor"
  )

## adjust the length of major ticks with the usual axis.ticks.length element
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_offset_minor"
  ) +
  scale_y_continuous(
    limits = c(10, 35),
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_offset_minor"
  ) +
  theme(
    axis.ticks.length = unit(10, "pt")
  )

## adjust the length of minor ticks with a new prism.ticks.length element
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_offset_minor"
  ) +
  scale_y_continuous(
    limits = c(10, 35),
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_offset_minor"
  ) +
  theme(
    axis.ticks.length = unit(10, "pt"),
    prism.ticks.length = unit(5, "pt")
  )

## to get log10 minor ticks just use a log10 scale and set the minor breaks
ggplot(msleep, aes(bodywt, brainwt)) +
  geom_point(na.rm = TRUE) +
  scale_x_log10(limits = c(1e0, 1e4),
                minor_breaks = rep(1:9, 4)*(10^rep(0:3, each = 9)),
                guide = "prism_offset_minor") +
  theme(axis.line = element_line(colour = "black"))

## change colour and tick length with the usual elements
base +
  scale_x_continuous(
    limits = c(0, 6),
    minor_breaks = seq(0, 6, 0.5),
    guide = "prism_offset_minor"
  ) +
```

```
  scale_y_continuous(
    limits = c(10, 35),
    minor_breaks = seq(10, 35, 1.25),
    guide = "prism_offset_minor"
  ) +
  theme(
    axis.ticks.length = unit(10, "pt"),
    prism.ticks.length = unit(5, "pt"),
    axis.ticks = element_line(colour = "red"),
    axis.line = element_line(colour = "blue")
  )
```

`preview_theme` *Preview Prism themes*

Description

Quickly generate a preview of a ggprism theme. See `names(ggprism_data$themes)` for valid palette names.

Usage

```
preview_theme(palette)
```

Arguments

`palette`	`string`. Palette name.

Value

Returns an object of class *ggplot*.

Examples

```
library(ggplot2)

## see names of available themes
names(ggprism_data$themes)

## preview a theme
preview_theme("floral")
```

prism_colour_pal *Prism colour palettes*

Description

A collection of colour palettes which mirror the colour schemes available in GraphPad Prism.

Usage

```
prism_colour_pal(palette = "colors")

prism_color_pal(palette = "colors")
```

Arguments

palette	`string`. Palette name, use `lengths(ggprism_data$colour_palettes)` to show all valid palette names and their number of values each palette supports.

Value

Returns a function which takes a single integer as its only argument and returns a character vector of hexadecimal colours. See the examples below for usage.

Examples

```
library(ggplot2)

## list all available colour palettes and their lengths
lengths(ggprism_data$colour_palettes)

## select some colours from a palette
prism_colour_pal(palette = "starry")(4)

## see all the colours in a specific palette
# define a function for convenience
library(scales)

show_palette <- function(palette) {
  scales::show_col(
    prism_colour_pal(palette = palette)(
      attr(prism_colour_pal(palette = palette), "max_n")
    )
  )
}

# show the colours in the palette "pearl"
show_palette("pearl")
```

`prism_fill_pal` *Prism fill palettes*

Description

A collection of fill palettes which mirror the colour schemes available in GraphPad Prism.

Usage

```
prism_fill_pal(palette = "colors")
```

Arguments

`palette`	`string`. Palette name, see `lengths(ggprism_data$fill_palettes)` for valid palette names.

Value

Returns a function which takes a single integer as its only argument and returns a character vector of hexadecimal colours. See the examples below for usage.

Examples

```
library(ggplot2)

## list all available fill palettes and their lengths
lengths(ggprism_data$fill_palettes)

## select some colours from a palette
prism_fill_pal(palette = "summer")(4)

## see all the colours in a specific palette
# define a function for convenience
library(scales)

show_palette <- function(palette) {
  scales::show_col(
    prism_fill_pal(palette = palette)(
      attr(prism_fill_pal(palette = palette), "max_n")
    )
  )
}

# show the colours in the palette "pearl"
show_palette("floral")
```

`prism_shape_pal` *Prism shape palettes*

Description

Shape palettes that approximate those used in GraphPad Prism. No unicode characters are used, only the default symbols available in R.

Usage

```
prism_shape_pal(palette = c("default", "filled", "complete"))
```

Arguments

`palette`	`string`. Palette name, one of: `default`, `filled`, or `complete`.

Details

The `default` palette supports up to 9 values. It does not use any symbols with a fill.

The `filled` palette supports up to 10 values. The first 5 symbols have a fill.

The `complete` palette supports up to 14 values. Symbols 5 to 9 have a fill.

Value

Returns a function which takes a single integer as its only argument and returns a character vector of integers which correspond to R plot pch symbols. See the examples below for usage.

Examples

```
library(ggplot2)

## list all available shape palettes
ggprism_data$shape_palettes

## select some shapes from a palette
prism_shape_pal(palette = "filled")(4)

## see all the shapes in a specific palette
# define a function for convenience
show_shapes <- function(palette) {
  df_shapes <- ggprism_data$shape_palettes[[palette]][, -1]
  df_shapes$pch_f <- factor(df_shapes$pch, levels = df_shapes$pch)

  ggplot(df_shapes, aes(x = 0, y = 0, shape = pch)) +
    geom_point(aes(shape = pch), size = 5, fill = 'red') +
    scale_shape_identity() +
    facet_wrap(~ pch_f) +
    theme_void()
}
```

```
# show the shapes in the palette "complete"
show_shapes("complete")
```

`scale_colour_prism` *Prism colour scales (discrete)*

Description

A collection of discrete colour scales that use palettes which mirror the colour schemes available in GraphPad Prism.

Usage

```
scale_colour_prism(palette = "colors", ...)

scale_color_prism(palette = "colors", ...)
```

Arguments

`palette`	`string`. Palette name, use `lengths(ggprism_data$colour_palettes)` to show all valid palette names and their number of values each palette supports.
`...`	Arguments passed on to `ggplot2::discrete_scale`

`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 (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

`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 values
- A character vector that defines possible values of the scale and their order

- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

`expand` For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

`na.translate` Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

`na.value` If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where `NA` is always placed at the far right.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` uses all the levels in the factor.

`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.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

Value

Returns a ggproto object of class *ScaleDiscrete* which works with *colour* aesthetics.

Examples

```
library(ggplot2)

## base plot
base <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(cyl))) +
  geom_point(size = 3)

## works pretty much the same as ggplot2 scale_colour_manual
base +
  scale_colour_prism(palette = "candy_bright")

## try combining the ggprism colour and fill scales
base2 <- ggplot(mpg, aes(x = class, y = hwy, fill = class, colour = class)) +
  geom_boxplot()

base2 +
  scale_fill_prism(palette = "floral") +
  scale_colour_prism(palette = "floral")

## change colour scale title in legend
base +
```

```
  scale_colour_prism(
    palette = "candy_bright",
    name = "Cylinders"
  )

## change colour labels in legend
base +
  scale_colour_prism(
    palette = "candy_bright",
    name = "Cylinders",
    label = c("4 cyl", "6 cyl", "8 cyl")
  )

## change colour labels in legend with a function
base +
  scale_colour_prism(
    palette = "candy_bright",
    name = "Cylinders",
    label = function(x) paste(x, "cyl")
  )

## change order of colours in legend
base +
  scale_colour_prism(
    palette = "candy_bright",
    name = "Cylinders",
    label = function(x) paste(x, "cyl"),
    breaks = c(8, 4, 6)
  )

## to change which colour is assigned to which cyl
## you need to change the factor levels in the underlying data
base <- ggplot(mtcars, aes(x = wt, y = mpg,
                           colour = factor(cyl, levels = c(6, 4, 8)))) +
  geom_point(size = 3)

base +
  scale_colour_prism(
    palette = "candy_bright",
    name = "Cylinders"
  )
```

`scale_fill_prism` *Prism fill scales (discrete)*

Description

A collection of discrete fill scales that use palettes which mirror the colour schemes available in GraphPad Prism.

Usage

```
scale_fill_prism(palette = "colors", ...)
```

Arguments

palette	string. Palette name, see lengths(ggprism_data$fill_palettes) for valid palette names.
...	Arguments passed on to ggplot2::discrete_scale

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 (the scale limits)
- A character vector of breaks
- A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

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 values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

expand For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function expansion() to generate the values for the expand argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

na.translate Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify na.translate = FALSE.

na.value If na.translate = TRUE, what aesthetic value should the missing values be displayed as? Does not apply to position scales where NA is always placed at the far right.

drop Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE uses all the levels in the factor.

`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.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

Value

Returns a ggproto object of class *ScaleDiscrete* which works with *fill* aesthetics.

Examples

```
library(ggplot2)

## base plot
base <- ggplot(mtcars, aes(x = mpg, fill = factor(cyl))) +
  geom_density(alpha = 0.75)

## works pretty much the same as ggplot2 scale_fill_manual
base +
  scale_fill_prism(palette = "candy_bright")

## try combining the ggprism colour and fill scales
base2 <- ggplot(mtcars, aes(x = mpg, fill = factor(cyl), colour = factor(cyl))) +
  geom_density(alpha = 0.75)

base2 +
  scale_fill_prism(palette = "floral") +
  scale_colour_prism(palette = "floral")

## change fill scale title in legend
base +
  scale_fill_prism(
    palette = "candy_bright",
    name = "Cylinders"
  )

## change fill labels in legend
base +
  scale_fill_prism(
    palette = "candy_bright",
    name = "Cylinders",
    label = c("4 cyl", "6 cyl", "8 cyl")
  )

## change fill labels in legend with a function
base +
  scale_fill_prism(
    palette = "candy_bright",
    name = "Cylinders",
    label = function(x) paste(x, "cyl")
```

```
  )

## change order of fills in legend
base +
  scale_fill_prism(
    palette = "candy_bright",
    name = "Cylinders",
    label = function(x) paste(x, "cyl"),
    breaks = c(8, 4, 6)
  )

## to change which fill is assigned to which cyl
## you need to change the factor levels in the underlying data
base <- ggplot(mtcars, aes(x = mpg,
                           fill = factor(cyl, levels = c(6, 4, 8)))) +
  geom_density(alpha = 0.75)

base +
  scale_fill_prism(
    palette = "candy_bright",
    name = "Cylinders"
  )
```

`scale_shape_prism` *Prism shape scales (discrete)*

Description

Shape scales that approximate those used in GraphPad Prism. No unicode characters are used, only the default symbols available in R.

Usage

```
scale_shape_prism(palette = "default", ...)
```

Arguments

`palette`	`string`. Palette name, one of: `default`, `filled`, or `complete`.
`...`	Arguments passed on to `ggplot2::discrete_scale`
	`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 (the scale limits)
	• A character vector of breaks
	• A function that takes the limits as input and returns breaks as output. Also accepts rlang lambda function notation.

`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 values
- A character vector that defines possible values of the scale and their order
- A function that accepts the existing (automatic) values and returns new ones. Also accepts rlang lambda function notation.

`expand` For position scales, a vector of range expansion constants used to add some padding around the data to ensure that they are placed some distance away from the axes. Use the convenience function `expansion()` to generate the values for the `expand` argument. The defaults are to expand the scale by 5% on each side for continuous variables, and by 0.6 units on each side for discrete variables.

`na.translate` Unlike continuous scales, discrete scales can easily show missing values, and do so by default. If you want to remove missing values from a discrete scale, specify `na.translate = FALSE`.

`na.value` If `na.translate = TRUE`, what aesthetic value should the missing values be displayed as? Does not apply to position scales where `NA` is always placed at the far right.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` uses all the levels in the factor.

`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.

`call` The `call` used to construct the scale for reporting messages.

`super` The super class to use for the constructed scale

Details

The `default` palette supports up to 9 values. It does not use any symbols with a fill.

The `filled` palette supports up to 10 values. The first 5 symbols have a fill.

The `complete` palette supports up to 14 values. Symbols 5 to 9 have a fill.

Value

Returns a ggproto object of class *ScaleDiscrete* which works with *shape* aesthetics.

Examples

```
library(ggplot2)

## list all available shape palettes
ggprism_data$shape_palettes

## define a base plot
base <- ggplot(mtcars, aes(x = wt, y = mpg,
                           shape = factor(cyl))) +
  geom_point(size = 3)

## works pretty much the same as ggplot2 scale_shape_manual
base +
  scale_shape_prism(palette = "complete")

## change shape scale title in legend
base +
  scale_shape_prism(
    palette = "default",
    name = "Cylinders"
  )

## change shape labels in legend
base +
  scale_shape_prism(
    palette = "default",
    name = "Cylinders",
    label = c("4 cyl", "6 cyl", "8 cyl")
  )

## change shape labels in legend with a function
base +
  scale_shape_prism(
    palette = "default",
    name = "Cylinders",
    label = function(x) paste(x, "cyl")
  )

## change order of shapes in legend
base +
  scale_shape_prism(
    palette = "default",
    name = "Cylinders",
    label = function(x) paste(x, "cyl"),
    breaks = c(8, 4, 6)
  )

## to change which shape is assigned to which cyl
## you need to change the factor levels in the underlying data
base <- ggplot(mtcars, aes(x = wt, y = mpg,
                           shape = factor(cyl, levels = c(6, 4, 8)))) +
  geom_point(size = 3)
```

```
base +
  scale_shape_prism(
    palette = "default",
    name = "Cylinders"
  )

## see all the shapes in a specific palette
# define a function for convenience
show_shapes <- function(palette) {
  df_shapes <- ggprism_data$shape_palettes[[palette]][, -1]
  df_shapes$pch_f <- factor(df_shapes$pch, levels = df_shapes$pch)

  ggplot(df_shapes, aes(x = 0, y = 0, shape = pch)) +
    geom_point(aes(shape = pch), size = 5, fill = 'red') +
    scale_shape_identity() +
    facet_wrap(~ pch_f) +
    theme_void()
}

# show the shapes in the palette "complete"
show_shapes("complete")
```

`theme_prism` *Prism themes*

Description

A collection of ggplot2 themes that use palettes which mirror the colour schemes available in GraphPad Prism.

Usage

```
theme_prism(
  palette = "black_and_white",
  base_size = 14,
  base_family = "sans",
  base_fontface = "bold",
  base_line_size = base_size/14,
  base_rect_size = base_size/14,
  axis_text_angle = 0,
  border = FALSE
)
```

Arguments

`palette`	`string`. Palette name, use `names(ggprism_data$themes)` to show all valid palette names.
`base_size`	`numeric`. Base font size, given in `"pt"`.

base_family	string. Base font family, default is "sans".
base_fontface	string. Base font face, default is "bold".
base_line_size	numeric. Base linewidth for line elements
base_rect_size	numeric. Base linewidth for rect elements
axis_text_angle	integer. Angle of axis text in degrees. One of: 0, 45, 90, 270.
border	logical. Should a border be drawn around the plot? Clipping will occur unless e.g. coord_cartesian(clip = "off") is used.

Value

Returns a list-like object of class *theme*.

Examples

```
library(ggplot2)

# see ?preview_theme for a convenient function to preview ggprism themes
# before using theme_prism

## base plot
base <- ggplot(mpg, aes(x = displ, y = cty, colour = class)) +
  geom_point()

## default palette is "black_and_white"
## default base_size is 14 (compared with 11 for theme_grey)
base +
  theme_prism()

## try some other palettes
base +
  theme_prism(palette = "office")

base +
  theme_prism(palette = "flames")

## try matching the theme_prism palette with same colour palette
base +
  theme_prism(palette = "stained_glass") +
  scale_color_prism(palette = "stained_glass")

base +
  theme_prism(palette = "candy_bright") +
  scale_color_prism(palette = "candy_bright")

## change the font face
base +
  theme_prism(base_fontface = "plain")

## change the font family
base +
```

```
  theme_prism(base_family = "serif")

## base_line_size scales automatically as you change base_size
base +
  theme_prism(base_size = 10)

## but you can also change it manually
base +
  theme_prism(base_size = 16, base_line_size = 0.8)

## easily change x axis text angle
base +
  theme_prism(axis_text_angle = 45)

## add a border (need to turn off clipping)
base +
  theme_prism(border = TRUE) +
  coord_cartesian(clip = "off")

## change border thickness
base +
  theme_prism(border = TRUE, base_rect_size = 2) +
  coord_cartesian(clip = "off")
```

`wings` *Wing morphology of mutant flies*

Description

Fold changes of different measures of wing morphology in heterozygous (Tps1MIC/+) and homozygous (Tps1MIC) Tps1 mutant flies. Data are expressed as percentage change relative to the mean of the heterozygous mutants.

Usage

```
wings
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 120 rows and 4 columns.

Details

40 flies were measured in total, with 3 measurements taken per fly.

sex `factor`. Male or female.

genotype `factor`. Heterozygous (Tps1MIC/+) or homozygous (Tps1MIC) mutant

measure `factor`. Type of wing measurement: wing size, cell size, or cell number

percent.change `double`. Value measured.

References

Matsushita, R, Nishimura, T. Trehalose metabolism confers developmental robustness and stability in *Drosophila* by regulating glucose homeostasis. Commun Biol 3, 170 (2020). doi:10.1038/s4200302008891

Index

∗ **datasets**
    ggprism_data, 12
    wings, 36

add_pvalue, 2
annotation_ticks, 9

coord_flip, *4*

element_text(), *13, 14*, *17*, *19*
expansion(), *27*, *29*, *32*

ggplot2::discrete_scale, *26*, *29*, *31*
ggprism_data, 12
glue, *3*
guide_axis, *14*
guide_prism_bracket, 12
guide_prism_minor, 14
guide_prism_offset, 17
guide_prism_offset_minor, 19
guides(), *27*, *30*, *32*

labs(), *13, 14*, *17*, *19*
lambda, *26, 27*, *29*, *31, 32*
layer, *4*

position_dodge, *4*
preview_theme, 22
prism_color_pal *(*prism_colour_pal*)*, 23
prism_colour_pal, 23
prism_fill_pal, 24
prism_shape_pal, 25

scale_color_prism *(*scale_colour_prism*)*, 26
scale_colour_prism, 26
scale_fill_prism, 28
scale_shape_prism, 31

theme, *15*, *20*
theme(), *13, 14*, *17*, *19*
theme_prism, 34

unit, *10*, *15*, *20*

waiver(), *13, 14*, *17*, *19*
wings, 36