Package: ggthemes (via r-universe)

October 22, 2026

Version 7.0.0.9000

Title Extra Themes, Scales and Geoms for 'ggplot2'

Depends R (>= 4.1.0)

Imports cli, ggplot2 (>= 3.5.2), graphics, grDevices, grid, lifecycle, methods, purrr, rlang, scales (>= 1.4.0), stats, tibble

Suggests dplyr, farver, lattice, maps, mapproj, pander, quantreg, spelling, systemfonts, testthat (>= 3.2.0), tidyr, vdiffr (>= 1.0.6), withr

Description Some extra themes, geoms, and scales for 'ggplot2'. Provides 'ggplot2' themes and scales that replicate the look of plots by Edward Tufte, Stephen Few, 'Fivethirtyeight', 'The Economist', 'Stata', 'Excel', and 'The Wall Street Journal', among others. Provides 'geoms' for Tufte's box plot and range frame.

License GPL-2

URL https://jrnold.github.io/ggthemes/, https://github.com/jrnold/ggthemes

BugReports https://github.com/jrnold/ggthemes/issues

LazyData true

Language en-US

Encoding UTF-8

Config/testthat/edition 3

Config/Needs/coverage covr

Config/Needs/data stringr

Config/Needs/lint lintr

Config/Needs/readme knitr, rmarkdown

Config/Needs/website pkgdown

Config/roxygen2/version 8.1.0

Roxygen list(markdown = TRUE)

Repository https://ar-puuk.r-universe.dev

Date/Publication 2026-10-21 23:19:23 UTC

RemoteUrl https://github.com/jrnold/ggthemes

RemoteRef HEAD

RemoteSha 4e81400562164e361b5d5d78c6ec4ca478c8bdec

Contents

bank_plot . . . 4
bank_plot_multiscale . . . 5
bank_slopes . . . 6
bank_slopes_multiscale . . . 9
calc_pal . . . 11
calc_shape_pal . . . 12
canva_pal . . . 13
canva_palettes . . . 14
circlefill_shape_pal . . . 15
cleveland_shape_pal . . . 16
colorblind_pal . . . 18
economist_pal . . . 20
economist_seq_pal . . . 21
excel_new_pal . . . 22
excel_pal . . . 23
extended_range_breaks_ . . . 23
few_pal . . . 25
few_shape_pal . . . 26
fivethirtyeight_pal . . . 27
gdocs_pal . . . 27
geom_rangeframe . . . 28
geom_tufteboxplot . . . 31
ggthemes_data . . . 34
hc_pal . . . 35
numbers_pal . . . 36
palette_pander . . . 37
ptol_pal . . . 38
scale_color_pander . . . 39
scale_colour_canva . . . 40
scale_colour_economist . . . 41
scale_colour_economist_c . . . 43
scale_colour_excel_new . . . 44
scale_colour_few . . . 46
scale_colour_fivethirtyeight . . . 48
scale_colour_gradient_tableau . . . 49
scale_colour_gradient2_tableau . . . 52
scale_colour_hc . . . 54
scale_colour_ptol . . . 55

scale_colour_stata . . . 57
scale_colour_tableau . . . 59
scale_colour_wsj . . . 61
scale_fill_calc . . . 62
scale_fill_excel . . . 64
scale_fill_gdocs . . . 66
scale_fill_numbers . . . 67
scale_fill_solarized . . . 69
scale_linetype_stata . . . 71
scale_shape_calc . . . 73
scale_shape_circlefill . . . 74
scale_shape_cleveland . . . 76
scale_shape_few . . . 78
scale_shape_stata . . . 79
scale_shape_tableau . . . 81
scale_shape_tremmel . . . 82
show_linetypes . . . 84
show_shapes . . . 85
smart_digits . . . 86
solarized_pal . . . 87
stat_fivenumber . . . 88
stata_linetype_pal . . . 90
stata_pal . . . 91
stata_shape_pal . . . 92
tableau_color_pal . . . 93
tableau_gradient_pal . . . 95
tableau_shape_pal . . . 96
theme_base . . . 97
theme_calc . . . 98
theme_clean . . . 99
theme_economist . . . 100
theme_excel . . . 102
theme_excel_new . . . 103
theme_few . . . 104
theme_fivethirtyeight . . . 105
theme_foundation . . . 105
theme_gdocs . . . 107
theme_hc . . . 107
theme_igray . . . 110
theme_map . . . 111
theme_numbers . . . 112
theme_pander . . . 112
theme_par . . . 114
theme_solarized . . . 115
theme_solid . . . 116
theme_stata . . . 117
theme_tufte . . . 119
theme_wsj . . . 120

tremmel_shape_pal 121
wsj_pal 122

Index **124**

`bank_plot` *Bank a plot's own data to 45 degrees*

Description

A convenience wrapper around `bank_slopes()` that extracts x/y directly from an already-specified `ggplot`, so you do not have to separately reconstruct the plotted vectors by hand. It builds `plot` with `ggplot2::ggplot_build()`, computes the banking ratio from one layer's fully resolved data (i.e. after stats, position adjustments, and faceting have been applied), and returns `plot + coord_fixed(ratio = ...)` (see `ggplot2::coord_fixed()`).

Usage

```
bank_plot(
  plot,
  method = c("ms", "as", "ao", "was"),
  cull = TRUE,
  layer = 1,
  ...
)
```

Arguments

`plot`	A `ggplot` object.
`method, cull, ...`	Passed to `bank_slopes()`.
`layer`	Integer. Which layer of `plot` to extract x/ y from. Defaults to the first layer.

Details

Segments are never averaged across a group or facet panel boundary: slopes are computed within each combination of `group` and `PANEL` and then combined, so a line plot with multiple series (or facets) is banked correctly rather than picking up spurious slopes between the end of one line and the start of the next.

Note that `ggplot2::coord_fixed()` applies a single ratio to every panel, so faceted plots are banked using the combined data from all panels rather than a ratio tailored to each one individually.

Value

The `plot`, with `ggplot2::coord_fixed()` added.

See Also

bank_slopes()

Examples

```
library("ggplot2")
x <- seq_along(sunspot.year)
y <- as.numeric(sunspot.year)
p <- ggplot(data.frame(x = x, y = y), aes(x = x, y = y)) +
  geom_line()
bank_plot(p)
```

---

bank_plot_multiscale *Bank a plot's own data at every scale*

Description

A convenience wrapper around bank_slopes_multiscale() that extracts y directly from an already-specified ggplot and returns one copy of the plot per scale of interest, each with the appropriate ggplot2::coord_fixed() applied. The result is the small-multiples display used throughout Heer and Agrawala (2006): the same data, banked to reveal trends at different frequencies.

Usage

```
bank_plot_multiscale(
  plot,
  method = c("ms", "as", "ao", "was"),
  cull = TRUE,
  layer = 1,
  ...
)
```

Arguments

plot	A ggplot object.
method, cull, ...	Passed to bank_slopes_multiscale().
layer	Integer. Which layer of plot to extract y from. Defaults to the first layer.

Details

Multi-scale banking is defined on the frequency domain of a single series sampled on a regular grid, so unlike bank_plot() this function requires the chosen layer to hold exactly one series with evenly spaced x values.

Value

A named `list` of `ggplot` objects, one per retained scale, in ascending order of frequency and named by frequency index.

References

Heer, Jeffrey and Maneesh Agrawala, 2006. "Multi-Scale Banking to 45." IEEE Transactions On Visualization And Computer Graphics 12(5).

See Also

`bank_slopes_multiscale()`, `bank_plot()`

Examples

```
library("ggplot2")

y <- as.numeric(sunspot.year)
p <- ggplot(data.frame(x = seq_along(y), y = y), aes(x = x, y = y)) +
  geom_line()

# One plot per scale of interest, named by frequency index.
plots <- bank_plot_multiscale(p)
names(plots)

## Low-frequency trend across sunspot cycles
plots[[1]]

## The individual 11-year cycles
plots[[2]]
```

---

bank_slopes *Bank slopes to 45 degrees*

Description

Calculate the optimal aspect ratio of a line graph by banking the slopes to 45 degrees as suggested by W.S. Cleveland. This maximizes the ability to visually differentiate differences in slope. This function will calculate the optimal aspect ratio for a line plot using any of the methods described in Heer and Agrawala (2006). In their review of the methods they suggest using median absolute slope banking (`"ms"`), which produces aspect ratios which are generally the median of the various methods provided here.

Usage

```
bank_slopes(
  x,
  y,
  cull = TRUE,
  weight = NULL,
  method = c("ms", "as", "ao", "was"),
  ...
)
```

Arguments

`x`	A numeric vector of x values.
`y`	A numeric vector of y values.
`cull`	If `TRUE` (the default), drop segments with a slope of 0 or `Inf` before banking.
`weight`	**[Deprecated]** No longer used, but kept for backwards compatibility.
`method`	The banking method, one of: • `"ms"`, the default: median absolute slope. • `"as"`: average absolute slope. • `"ao"`: average absolute orientation. • `"was"`: weighted average absolute orientation.
`...`	**[Deprecated]** No longer used, but kept for backwards compatibility.

Value

A number, the aspect ratio `y / x`, for use as `ratio` in `ggplot2::coord_fixed()`.

Methods

As written, all of these methods calculate the aspect ratio (x /y), but `bank_slopes()` will return (y / x) to be compatible with `ggplot2::coord_fixed()`.

Median Absolute Slopes Banking

Let the aspect ratio be $\alpha = \frac{w}{h}$ then the median absolute slop banking is the α such that,

$$median \left| \frac{s_i}{\alpha} \right| = 1$$

Let $R_z = z_{max} - z_{min}$ for $z = x, y$, and $M = median \|s_i\|$. Then,

$$\alpha = M \frac{R_x}{R_y}$$

Average Absolute Slope Banking

Let the aspect ratio be $\alpha = \frac{w}{h}$. then the mean absolute slope banking is the α such that,

$$mean \left| \frac{s_i}{\alpha} \right| = 1$$

Average Absolute Orientation Banking

Rather than averaging the slopes themselves, this method averages the *orientation* (angle) of each segment, since perceived slope differences are more closely related to angle than to the raw ratio dy/dx. Let $s'_i = s_i R_x / R_y$ be the range-normalized slopes. Then α is chosen such that,

$$mean \left| \arctan\left(\frac{s'_i}{\alpha}\right) \right| = \frac{\pi}{4}$$

This has no closed-form solution and is found numerically with `stats::uniroot()`.

Weighted Average Absolute Orientation Banking

This is the weighted version of Average Absolute Orientation Banking from Heer and Agrawala (2006). Each segment's absolute orientation is weighted by its length in display space, so both the orientation and its weight depend on α. With s'_i as above and segment run dx_i, α is chosen such that,

$$\frac{\sum_i |\arctan(s'_i/\alpha)|\, dx_i \sqrt{1 + (s'_i/\alpha)^2}}{\sum_i dx_i \sqrt{1 + (s'_i/\alpha)^2}} = \frac{\pi}{4}$$

This has no closed-form solution and is found numerically with `stats::uniroot()`.

All of these methods consider the entirety of the data at once, so they accentuate local features and can obscure larger-scale trends. Heer and Agrawala (2006) address this with multi-scale banking, which uses spectral analysis to identify the frequency scales present in the data and banks each one separately; see `bank_slopes_multiscale()` and `bank_plot_multiscale()`.

References

Cleveland, W. S., M. E. McGill, and R. McGill. The Shape Parameter of a Two-Variable Graph. Journal of the American Statistical Association, 83:289-300, 1988

Heer, Jeffrey and Maneesh Agrawala, 2006. 'Multi-Scale Banking to 45' IEEE Transactions On Visualization And Computer Graphics.

Cleveland, W. S. 1993. 'A Model for Studying Display Methods of Statistical Graphs.' Journal of Computational and Statistical Graphics.

Cleveland, W. S. 1994. The Elements of Graphing Data, Revised Edition.

See Also

`lattice::banking()`, `bank_plot()` to bank a `ggplot` using its own data, and `bank_slopes_multiscale()` to bank each frequency scale in the data separately.

Examples

```
library("ggplot2")

# Use the classic sunspot data from Cleveland's original paper
x <- seq_along(sunspot.year)
y <- as.numeric(sunspot.year)
# Without banking
m <- ggplot(data.frame(x = x, y = y), aes(x = x, y = y)) +
  geom_line()
```

```
m

## Using the default method, Median Absolute Slope
ratio <- bank_slopes(x, y)
m + coord_fixed(ratio = ratio)

## Average Absolute Slope
m + coord_fixed(ratio = bank_slopes(x, y, method = "as"))

## Average Absolute Orientation
m + coord_fixed(ratio = bank_slopes(x, y, method = "ao"))

## Weighted Average Absolute Slope: each segment is weighted by its run in
## x, so this only differs from "as" when x is not evenly spaced
m + coord_fixed(ratio = bank_slopes(x, y, method = "was"))

## Culling removes slopes of 0 or Inf before banking, which matters when
## the data contains runs of repeated x or y values
bank_slopes(x, y, cull = TRUE)
```

`bank_slopes_multiscale`

Multi-scale banking to 45 degrees

Description

Compute a set of aspect ratios, one per frequency scale present in a series, using the multi-scale banking algorithm of Heer and Agrawala (2006). Single-scale banking (`bank_slopes()`) considers the whole series at once, so it accentuates local features and can obscure larger-scale trends. Multi-scale banking instead uses spectral analysis to find the scales that carry real energy, low-pass filters the data to each of those scales in turn, and banks the resulting trend curve, yielding one aspect ratio per scale.

Usage

```
bank_slopes_multiscale(
  y,
  method = c("ms", "as", "ao", "was"),
  cull = TRUE,
  window = 3,
  sd = 1,
  threshold = NULL,
  scale_factor = 1.25
)
```

Arguments

`y`	`numeric` series of evenly spaced observations.

`method`, `cull`	Passed to `bank_slopes()`. The defaults are those Heer and Agrawala used for all results reported in their Section 3.2: median absolute slope banking with slopeless line culling.
`window`	`integer` width, in frequency bins, of the Gaussian kernel used to smooth the power spectrum.
`sd`	`numeric` standard deviation of that Gaussian kernel.
`threshold`	`numeric` power above which a frequency bin counts as a scale of interest. Defaults to the mean of the smoothed power spectrum. Raise it to select fewer scales.
`scale_factor`	`numeric` minimum ratio between successive retained aspect ratios.

Details

The procedure is Algorithm 1 of Heer and Agrawala (2006):

1. Take the discrete Fourier transform of `y` and form the power spectrum from the squared coefficient magnitudes.
2. Smooth the spectrum by convolving it with a Gaussian kernel, since spectral energy tends to arrive in "clumps" containing local oscillation.
3. Threshold the smoothed spectrum. Contiguous runs above the threshold are collapsed to their highest-frequency bin, capturing the total contribution of that region of energy.
4. For each retained scale, low-pass filter `y` to remove all higher frequencies and bank the resulting trend curve to 45 degrees using `bank_slopes()`.
5. Discard aspect ratios within `scale_factor` of the previous retained ratio, since they would produce visually redundant charts.

The scale corresponding to the data in its entirety is always included.

Because the algorithm is defined on the frequency domain of `y` alone, it assumes observations are evenly spaced in `x`; the banking of each trend curve uses `x = seq_along(y)`.

Value

A `tibble::tibble()` with one row per retained scale, in ascending order of frequency, and columns:

- `frequency`: `integer` frequency index, i.e. the number of times the trend repeats across the series.
- `ratio`: `numeric` aspect ratio in the `y / x` sense used by `ggplot2::coord_fixed()`.
- `aspect_ratio`: `numeric` the same value as width / height, the convention in which the banking literature reports aspect ratios.

References

Heer, Jeffrey and Maneesh Agrawala, 2006. "Multi-Scale Banking to 45." IEEE Transactions On Visualization And Computer Graphics 12(5).

Cleveland, W. S. 1993. "A Model for Studying Display Methods of Statistical Graphs." Journal of Computational and Statistical Graphics.

See Also

bank_slopes() for single-scale banking, and bank_plot_multiscale() to bank a ggplot at every scale.

Examples

```
library("ggplot2")

# Sunspot activity, the classic example from Cleveland and from Heer and
# Agrawala's Section 3.2.1. Spectral analysis identifies scales at frequency
# indices 7, 10, 31 and 36, plus the data in its entirety; culling similar
# aspect ratios leaves two charts worth drawing.
y <- as.numeric(sunspot.year)
bank_slopes_multiscale(y)

# `ratio` is ready for coord_fixed(); `aspect_ratio` is the same value as
# width / height, the convention used in the banking literature.
scales <- bank_slopes_multiscale(y)

m <- ggplot(data.frame(x = seq_along(y), y = y), aes(x = x, y = y)) +
  geom_line()

## The low-frequency trend: the oscillation of high points across cycles
m + coord_fixed(ratio = scales$ratio[[1]])

## The 11-year cycle: a steep onset followed by a more gradual decay
m + coord_fixed(ratio = scales$ratio[[2]])

## Raise the threshold to select fewer scales
bank_slopes_multiscale(y, threshold = Inf)

## Any of the single-scale banking methods can be used for each scale
bank_slopes_multiscale(y, method = "ao")
```

---

calc_pal *Calc color palette (discrete)*

Description

Color palettes from LibreOffice Calc. This palette has 12 values.

Usage

```
calc_pal()
```

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

See Also

Other color calc: scale_fill_calc()

Examples

```
library("scales")

show_col(calc_pal()(12))
```

---

calc_shape_pal *Calc shape palette (discrete)*

Description

Shape palette based on the shapes used in LibreOffice Calc.

Usage

```
calc_shape_pal(unicode = FALSE)
```

Arguments

unicode	If TRUE, return pch codes derived from Unicode glyphs, as this palette did before ggthemes 6.1.0. Glyph shapes are drawn by the device font, so they render as blank boxes in a font without coverage; the default returns base pch codes, which every font can draw.

Value

A palette function. It takes the number of shapes n and returns an integer vector of n shape (pch) codes, and can be used as the palette argument of ggplot2::discrete_scale().

Note

This palette supports seven values by default and thirteen with unicode = TRUE. Six of Calc's thirteen symbols – the solid down, left and right triangles, the bowtie, the hourglass and the four-pointed star – have no base pch equivalent and are dropped rather than approximated by a different shape. Restoring them with unicode = TRUE needs a font covering Geometric Shapes, Dingbats and Miscellaneous Mathematical Symbols-B; Noto Sans Symbols 2 is effectively the only free font with the last of these.

See Also

Other shapes calc: scale_shape_calc()

Examples

```
# The seven shapes with a font-independent equivalent.
show_shapes(calc_shape_pal()(7))

## Not run:
  # All thirteen, drawn from the device font. Needs a font covering Geometric
  # Shapes, Dingbats and Miscellaneous Mathematical Symbols-B, such as
  # Noto Sans Symbols 2.
  show_shapes(calc_shape_pal(unicode = TRUE)(13))

## End(Not run)
```

---

canva_pal *Canva.com color palettes*

Description

150+ color palettes from canva.com. See canva_palettes.

Usage

```
canva_pal(palette = "Fresh and bright")
```

Arguments

palette	Palette name. See the names of canva_palettes for valid names.

Value

A function that takes a single value, the number of colors to use.

See Also

Other color canva: `scale_colour_canva()`

Examples

```
require("ggplot2")
require("tibble")
if (require("purrr") && require("scales") && require("dplyr")) {
  canva_df <- map2_df(
    canva_palettes,
    names(canva_palettes),
    ~ tibble(
      colors = .x,
      .id = seq_along(colors),
      palette = .y
    )
  )
```

```
  ggplot(
    canva_df,
    aes(
      y = palette,
      x = .id,
      fill = colors
    )
  ) +
    geom_raster() +
    scale_fill_identity(guide = FALSE) +
    theme_minimal() +
    theme(
      panel.grid = element_blank(),
      axis.text.x = element_blank()
    ) +
    labs(x = "", y = "")

  show_col(canva_pal("Fresh and bright")(4))
  show_col(canva_pal("Cool blues")(4))
  show_col(canva_pal("Modern and crisp")(4))
}
```

`canva_palettes` *150 color palettes from Canva*

Description

150 four-color palettes by the canva.com design school. These palettes were derived from photos and "impactful websites".

Usage

```
canva_palettes
```

Format

A named `list` of character vectors. The names are the palette names. The values of the character vectors are hex colors, e.g. `"#f98866"`.

References

- Janie Kliever, 100 Brilliant Color Combinations and How to Apply Them to Your Designs, *Canva.com*, June 20, 2015.
- Mary Stribley, Website Color Schemes: The Palettes of 50 Visually Impactful Websites to Inspire You, *Canva.com*, January 26, 2016.
- Schwabish, Jonathan. 150+ Color Palettes for Excel, *PolicyViz*, January 12, 2017.

Examples

```
require("ggplot2")
require("tibble")
if (require("purrr") && require("scales") && require("dplyr")) {
  canva_df <- map2_df(
    canva_palettes,
    names(canva_palettes),
    ~ tibble(
      colors = .x,
      .id = seq_along(colors),
      palette = .y
    )
  )
  ggplot(
    canva_df,
    aes(
      y = palette,
      x = .id,
      fill = colors
    )
  ) +
    geom_raster() +
    scale_fill_identity(guide = FALSE) +
    theme_minimal() +
    theme(
      panel.grid = element_blank(),
      axis.text.x = element_blank()
    ) +
    labs(x = "", y = "")

  show_col(canva_pal("Fresh and bright")(4))
  show_col(canva_pal("Cool blues")(4))
  show_col(canva_pal("Modern and crisp")(4))
}
```

`circlefill_shape_pal` *Filled Circle Shape palette (discrete)*

Description

[Deprecated]

This function was deprecated because unicode glyphs used for the circles vary in size, making them unusable for plotting.

Shape palette with circles varying by amount of fill. This uses the set of 3 circle fill values in Lewandowsky and Spence (1989): solid, hollow, half-filled, with two additional fill amounts: three-quarters, and one-quarter.

This palette supports up to five values.

Usage

```
circlefill_shape_pal()
```

Value

A palette function. It takes the number of shapes `n` and returns an integer vector of `n` shape (`pch`) codes, and can be used as the `palette` argument of `ggplot2::discrete_scale()`.

References

Lewandowsky, Stephan and Ian Spence (1989) "Discriminating Strata in Scatterplots", Journal of the American Statistical Association, https://www.jstor.org/stable/2289649

See Also

Other shapes: `cleveland_shape_pal()`, `scale_shape_circlefill()`, `scale_shape_cleveland()`, `scale_shape_tremmel()`, `tremmel_shape_pal()`

Examples

```
circlefill_shape_pal()(3)
```

---

cleveland_shape_pal *Shape palette from Cleveland's "Elements of Graphing Data"*

Description

Shape palettes for overlapping and non-overlapping points.

Usage

```
cleveland_shape_pal(overlap = TRUE, unicode = FALSE)
```

Arguments

`overlap`	`logical` Use the scale for overlapping points?
`unicode`	If `TRUE`, return pch codes derived from Unicode glyphs, as this palette did before ggthemes 6.1.0. Glyph shapes are drawn by the device font, so they render as blank boxes in a font without coverage; the default returns base pch codes, which every font can draw.

Value

A palette function. It takes the number of shapes `n` and returns an integer vector of `n` shape (`pch`) codes, and can be used as the `palette` argument of `ggplot2::discrete_scale()`.

Note

In the *Elements of Graphing Data*, W.S. Cleveland suggests two shape palettes for scatter plots: one for overlapping data and another for non-overlapping data. The symbols for overlapping data rely on pattern discrimination, while the symbols for non-overlapping data vary the amount of fill.

Following Tremmel (1995), the circle with a vertical line is replaced by an encircled plus sign.

`cleveland_shape_pal(overlap = TRUE)` supports four values on either branch.

`cleveland_shape_pal(overlap = FALSE)` supports three values by default and five with `unicode = TRUE`. Its five source symbols encode *fill fraction*, which base pch cannot express, so the two partially filled circles are dropped rather than approximated by a different shape. To encode a proportion, map `alpha` or `fill` instead; to restore the five glyphs, use `unicode = TRUE` with a font that covers Mathematical Operators, such as STIX Two Text.

The truncation is arguably an improvement. Tremmel (1995) Experiment 2 tested exactly this five-symbol set and found the fill-graded circles the worst performers measured, with the encircled plus and encircled dot the slowest pair and the one producing the most errors. The three shapes that survive are the better-separating subset.

References

Cleveland WS. *The Elements of Graphing Data*. Revised Edition. Hobart Press, Summit, NJ, 1994, pp. 154-164, 234-239.

Tremmel, Lothar, (1995) "The Visual Separability of Plotting Symbols in Scatterplots", *Journal of Computational and Graphical Statistics*, https://www.jstor.org/stable/1390760

See Also

Other shapes: `circlefill_shape_pal()`, `scale_shape_circlefill()`, `scale_shape_cleveland()`, `scale_shape_tremmel()`, `tremmel_shape_pal()`

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, shape = factor(gear))) +
  facet_wrap(~am) +
  theme_bw()
# overlapping symbol palette
p + scale_shape_cleveland()
# non-overlapping symbol palette
p + scale_shape_cleveland(overlap = FALSE)
```

colorblind_pal *Colorblind color palette (discrete) and scales*

Description

An eight-color colorblind safe qualitative discrete palette.

[Deprecated] `scale_colour_colorblind()` is deprecated in favor of `scale_colour_colourblind()` and `scale_color_colorblind()`.

Usage

```
colorblind_pal(black = TRUE)

colourblind_pal(black = TRUE)

scale_colour_colourblind(black = TRUE, ...)

scale_colour_colorblind(black = TRUE, ...)

scale_color_colorblind(black = TRUE, ...)

scale_fill_colourblind(black = TRUE, ...)

scale_fill_colorblind(black = TRUE, ...)
```

Arguments

`black` If `FALSE`, drop black from the palette. Black is often used elsewhere in a figure (e.g. text, axes), so including it as a data color can wrongly suggest that group is a default or baseline.

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

`colorblind_pal()` and `colourblind_pal()` return a palette function that takes the number of colors `n` and returns a character vector of `n` hex colors. The `scale_*()` functions return a ggplot2 scale object.

References

Chang, W. "Cookbook for R"

https://jfly.iam.u-tokyo.ac.jp/color/

See Also

The dichromat package, `scales::dichromat_pal()`, and `scale_color_tableau()` for other colorblind palettes.

Examples

```
library("ggplot2")
library("scales")

show_col(colorblind_pal()(8))
show_col(colorblind_pal(black = FALSE)(7))
p <- ggplot(mtcars) +
  geom_point(aes(
    x = wt,
    y = mpg,
    colour = factor(gear)
  )) +
  facet_wrap(~am)
p + theme_igray() + scale_colour_colourblind()
```

---

economist_pal *Economist color palette (discrete)*

Description

The classic *The Economist* chart palette: blues, grays, and greens, chosen and ordered for the number of colors requested. Red is not included in these palettes; *The Economist* reserves it to mark important data.

Usage

```
economist_pal(fill = TRUE)
```

Arguments

fill	If TRUE (the default), use the fill palette; otherwise, use the line palette. The two palettes choose and order the colors differently.

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

See Also

Other color economist: economist_seq_pal(), scale_colour_economist(), scale_colour_economist_c()

Examples

```
library("scales")

show_col(economist_pal()(6))
## fill palette
show_col(economist_pal(fill = TRUE)(6))
```

economist_seq_pal *Economist sequential color palettes*

Description

The "equal lightness colour scales" of *The Economist visual styleguide* (v1.2, 4 May 2017): six ordered steps for each of nine hues, running darkest to lightest. Use them for ordered data. They come from the paper's 2017 chart design, not the classic design of `economist_pal()` and `theme_economist()`.

Usage

```
economist_seq_pal(hue = "blue")

economist_gradient_pal(hue = "blue")
```

Arguments

hue	A string, the hue of the color scale. One of `"blue"` (the default), `"cyan"`, `"green"`, `"yellow"`, `"olive"`, `"purple"`, `"gold"`, `"gray"`, or `"red"`.

Details

`economist_seq_pal()` returns the six steps themselves, for discrete ordered data. `economist_gradient_pal()` interpolates between them, for continuous data.

Value

`economist_seq_pal()` returns a palette function that takes the number of colors `n` and returns the first `n` of the six hex colors for `hue`, for use with `ggplot2::discrete_scale()`. `economist_gradient_pal()` returns a palette function that takes a numeric vector `x` of values between 0 and 1 and returns hex colors interpolated between those steps, for use with `ggplot2::continuous_scale()`.

See Also

Other color economist: `economist_pal()`, `scale_colour_economist()`, `scale_colour_economist_c()`

Examples

```
library("scales")

# the six steps of one hue, darkest first
show_col(economist_seq_pal("blue")(6))
show_col(economist_seq_pal("red")(6))

# interpolated for continuous data
show_col(economist_gradient_pal("green")(seq(0, 1, length.out = 10)))
```

excel_new_pal *Excel (current versions) color palettes (discrete)*

Description

Color palettes used by current versions of Microsoft Office and Excel.

Usage

```
excel_new_pal(theme = "Office")
```

Arguments

theme	The name of the Office theme or color theme (not to be confused with ggplot2 themes) from which to derive the color palette. Available themes include: "Atlas", "Badge", "Berlin", "Celestial", "Crop", "Depth", "Droplet", "Facet", "Feathered", "Gallery", "Headlines", "Integral", "Ion Boardroom", "Ion", "Madison", "Main Event", "Mesh", "Office", "Office 2013", "Organic", "Parallax", "Parcel", "Retrospect", "Savon", "Slice", "Vapor Trail", "View", "Wisp", "Wood Type", "Aspect", "Blue Green", "Blue II", "Blue Warm", "Blue", "Grayscale", "Green Yellow", "Green", "Marquee", "Median", "Office 2007", "Orange Red", "Orange", "Paper", "Red Orange", "Red Violet", "Red", "Slipstream", "Violet II", "Violet", "Yellow Orange", "Yellow".

Details

In 2023 Microsoft replaced the long-standing Office theme with a new default and renamed the old one. The default here, "Office", is the current palette; "Office 2013" is the palette Excel used from 2013 until 2022. The former ggthemes names "Office Theme" and "Office 2007-2010" still work, and select "Office 2013" and "Office 2007" respectively.

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

See Also

Other color excel: excel_pal(), scale_colour_excel_new(), scale_fill_excel()

Examples

```
library("scales")

for (i in names(ggthemes::ggthemes_data$excel$themes)) {
  show_col(excel_new_pal(theme = i)(6))
}
```

excel_pal *Excel 97 ugly color palettes (discrete)*

Description

The color palettes used in Microsoft Excel 97 (and up until Excel 2007). Use this for that classic ugly look and feel. For ironic purposes only. 3D bars and pies not included. Please never use this color palette.

Usage

```
excel_pal(line = TRUE)
```

Arguments

line	If TRUE, use the palette for lines and points. Otherwise, use the palette for area.

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

See Also

Other color excel: excel_new_pal(), scale_colour_excel_new(), scale_fill_excel()

Examples

```
library("scales")

show_col(excel_pal()(7))
show_col(excel_pal(line = FALSE)(7))
```

extended_range_breaks_ *Pretty axis breaks inclusive of extreme values*

Description

This function returns pretty axis breaks that always include the extreme values of the data. This works by calling the extended Wilkinson algorithm (Talbot et al., 2010), constrained to solutions interior to the data range. Then, the minimum and maximum labels are moved to the minimum and maximum of the data range.

Usage

```
extended_range_breaks_(
  dmin,
  dmax,
  n = 5,
  Q = c(1, 5, 2, 2.5, 4, 3),
  w = c(0.25, 0.2, 0.5, 0.05)
)

extended_range_breaks(n = 5, ...)
```

Arguments

dmin	Minimum of the data range.
dmax	Maximum of the data range.
n	Desired number of breaks.
Q	Set of nice numbers.
w	Weights applied to the four optimization components (simplicity, coverage, density, and legibility).
...	Other arguments passed to extended_range_breaks_().

Details

extended_range_breaks_() implements the algorithm and returns the break values. extended_range_breaks() uses the conventions of the scales package, and returns a function.

Note that ggplot2 hands a breaks function the expanded scale limits, not the data range. Passing extended_range_breaks() directly to breaks therefore places the end labels at the panel edges rather than at the extremes of the data. To label the extremes, which is what pairs with geom_rangeframe(), apply the function to the data instead: scale_x_continuous(breaks = extended_range_breaks()(

Value

For extended_range_breaks_, the vector of axis label locations. For extended_range_breaks, a function which takes a single argument, a vector of data, and returns the vector of axis label locations.

Author(s)

Justin Talbot jtalbot@stanford.edu, Jeffrey B. Arnold, Baptiste Auguie

References

Talbot, J., Lin, S., Hanrahan, P. (2010) An Extension of Wilkinson's Algorithm for Positioning Tick Labels on Axes, InfoVis 2010.

See Also

Other tufte: geom_rangeframe(), geom_tufteboxplot(), theme_tufte()

Examples

```
# Pretty breaks that always include the extremes of the data
extended_range_breaks_(min(mtcars$wt), max(mtcars$wt))

# A function, in the style of the scales package
extended_range_breaks()(mtcars$wt)

library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  geom_rangeframe() +
  scale_x_continuous(breaks = extended_range_breaks()(mtcars$wt)) +
  scale_y_continuous(breaks = extended_range_breaks()(mtcars$mpg)) +
  theme_tufte()
```

---

few_pal *Color palettes from Few's "Show Me the Numbers"*

Description

Qualitative color palettes from Stephen Few (2012) *Show Me the Numbers*. There are three palettes: Light, Medium, and Dark. Each palette comprises nine colors: gray, blue, orange, green, pink, brown, purple, yellow, red. For n = 1, gray is used. For n > 1, the eight non-gray colors are used.

Usage

```
few_pal(palette = "Medium")
```

Arguments

palette	One of "Light", "Medium", "Dark".

Details

Use the light palette for filled areas, such as bar charts. Use the medium palette for points and lines. Use the dark palette for highlighting specific points or for small and thin lines and points.

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

References

Few, S. (2012) *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. 2nd edition. Analytics Press.

"Practical Rules for Using Color in Charts".

See Also

Other color few: scale_colour_few()

Examples

```
library("scales")

show_col(few_pal()(7))
show_col(few_pal("Dark")(7))
show_col(few_pal("Light")(7))
```

---

few_shape_pal *Shape palette from "Show Me the Numbers" (discrete)*

Description

Shape palette from Stephen Few's "Show Me the Numbers". The shape palette consists of five shapes: circle, square, triangle, plus, times.

Usage

```
few_shape_pal()
```

Value

A palette function. It takes the number of shapes n and returns an integer vector of n shape (pch) codes, and can be used as the palette argument of ggplot2::discrete_scale().

References

Few, S. (2012) *Show Me the Numbers: Designing Tables and Graphs to Enlighten*, Analytics Press, p. 208.

See Also

Other shapes few: scale_shape_few()

Examples

```
few_shape_pal()(5)

show_shapes(few_shape_pal()(5))
```

fivethirtyeight_pal *FiveThirtyEight color palette*

Description

The standard three-color FiveThirtyEight palette for line plots comprises blue, red, and green.

Usage

```
fivethirtyeight_pal()
```

Value

A palette function. It takes the number of colors `n` and returns a character vector of `n` hex colors, and can be used as the `palette` argument of `ggplot2::discrete_scale()`.

See Also

Other color fivethirtyeight: `scale_colour_fivethirtyeight()`

Examples

```
library("scales")

show_col(fivethirtyeight_pal()(3))
```

gdocs_pal *Google Docs color palette (discrete)*

Description

Color palettes from Google Docs. This palette includes 20 colors.

Usage

```
gdocs_pal()
```

Value

A palette function. It takes the number of colors `n` and returns a character vector of `n` hex colors, and can be used as the `palette` argument of `ggplot2::discrete_scale()`.

See Also

Other color gdocs: `scale_fill_gdocs()`

Examples

```
library("scales")

show_col(gdocs_pal()(24))
```

---

geom_rangeframe *Range frames*

Description

Axis lines which extend to the maximum and minimum of the plotted data.

Usage

```
geom_rangeframe(
  mapping = NULL,
  data = NULL,
  stat = "identity",
  position = "identity",
  ...,
  sides = "bl",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetic mappings created by aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer. There are three options: If NULL, the default, the data is inherited from the plot data as specified in the call to ggplot(). A data.frame, or other object, will override the plot data. All objects will be fortified to produce a data frame. See fortify() for which variables will be created. A function will be called with a single argument, the plot data. The return value must be a data.frame, and will be used as the layer data. A function can be created from a formula (e.g. ~ head(.x, 10)).
stat	The statistical transformation to use on the data for this layer. When using a geom_*() function to construct a layer, the stat argument can be used to override the default coupling between geoms and stats. The stat argument accepts the following: • A Stat ggproto subclass, for example StatCount.

- A string naming the stat. To give the stat as a string, strip the function name of the `stat_` prefix. For example, to use `stat_count()`, give the stat as `"count"`.
- For more information and other ways to specify the stat, see the layer stat documentation.

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

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

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

`sides` A string that controls which sides of the plot the frames appear on. It can be set to a string containing any of `"t"`, `"r"`, `"b"`, and `"l"` (for example, `"trbl"`), for top, right, bottom, and left. Any other value is an error: a frame cannot be drawn on a side that was not named, so a typo would otherwise silently draw nothing.

`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. To

include legend keys for all levels, even when no data exists, use TRUE. If NA, all levels are shown in legend, but unobserved levels are omitted.

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. annotation_borders().

Details

This should be used with coord_cartesian(clip="off") in order to correctly draw the lines.

Secondary axes (ggplot2::sec_axis()) only relabel the existing axis; they do not introduce a separate data range. Because of this, sides = "trbl" already draws correctly-positioned frames on the top/right edges of a panel that has a secondary axis – there is no separate "secondary" range for geom_rangeframe() to draw against.

Value

A ggplot2 layer.

Aesthetics

- colour
- size
- linetype
- alpha

References

Tufte, Edward R. (2001) The Visual Display of Quantitative Information, Chapter 6.

See Also

Other geom tufte: geom_tufteboxplot()

Other tufte: extended_range_breaks_(), geom_tufteboxplot(), theme_tufte()

Examples

```
library("ggplot2")

ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  geom_rangeframe() +
  coord_cartesian(clip = "off") +
  theme_tufte()

# In the example above,
# `coord_cartesian(clip="off")`` ensures that the full width of the line is drawn.
# if you know a better way to fix this,
# please open an issue or PR on github https://github.com/jrnold/ggthemes/issue
```

```
# sides = "trbl" also works with a secondary axis: the secondary axis only
# relabels the existing scale, so the frame is still correctly positioned.
ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  geom_rangeframe(sides = "trbl") +
  scale_y_continuous(sec.axis = sec_axis(~ . * 0.4251, name = "km/L")) +
  coord_cartesian(clip = "off") +
  theme_tufte()
```

`geom_tufteboxplot` *Tufte's box plot*

Description

Edward Tufte's revisions of the box plot as described in *The Visual Display of Quantitative Information*. This function provides several box plot variants:

- A point indicating the median, a gap indicating the interquartile range, and lines for whiskers.
- An offset line indicating the interquartile range and a gap indicating the median.
- A line indicating the interquartile range, a gap indicating the median, and points indicating the minimum and maximum values.
- A wide line indicating the interquartile range, a gap indicating the median, and lines indicating the minimum and maximum.

Usage

```
geom_tufteboxplot(
  mapping = NULL,
  data = NULL,
  stat = "fivenumber",
  position = "dodge",
  outlier.colour = "black",
  outlier.shape = 19,
  outlier.size = 1.5,
  outlier.stroke = 0.5,
  voffset = 0.01,
  hoffset = 0.005,
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE,
  median.type = "point",
  whisker.type = "line",
  ...
)
```

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, as a string. The default (`stat = 'fivenumber'`) calls `stat_fivenumber()` and produces whiskers that extend from the interquartile range to the extremes of the data; specifying `ggplot2::stat_boxplot()` will produce a more traditional boxplot with whiskers extending to the most extreme points that are < 1.5 IQR away from the hinges (i.e., the first and third quartiles).

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`outlier.colour` Color of outlying points.

`outlier.shape` Shape of outlying points.

`outlier.size` Size of outlying points.

`outlier.stroke` Stroke width of outlying points.

`voffset` Controls the size of the gap in the line representing the median when `median.type = "line"`. This is a fraction of the range of `y`.

`hoffset` Controls how much the interquartile line is offset from the whiskers when `median.type = "line"`. This is a fraction of the range of `x`.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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. `annotation_borders()`.

`median.type` How to represent the median, one of:

- `"point"`, the default: the median is a point, and the interquartile range is a gap in the line.
- `"line"`: the interquartile range is a line, possibly offset, and the median is a gap in the line.

`whisker.type` How to represent the whiskers, one of:

- `"line"`, the default: whiskers are lines.
- `"point"`: whiskers are points at `ymin` and `ymax`.

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

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

Value

A ggplot2 layer.

Aesthetics

- `x` (required)
- `y` (required)
- `colour`
- `size`
- `linewidth`

- linetype
- shape
- fill
- alpha

References

Tufte, Edward R. (2001) The Visual Display of Quantitative Information, Chapter 6.

McGill, R., Tukey, J. W. and Larsen, W. A. (1978) Variations of box plots. The American Statistician 32, 12-16.

See Also

ggplot2::geom_boxplot()

Other geom tufte: geom_rangeframe()

Other tufte: extended_range_breaks_(), geom_rangeframe(), theme_tufte()

Examples

```
library("ggplot2")

p <- ggplot(mtcars, aes(factor(cyl), mpg))
# with a point for the median and lines for whiskers
p + geom_tufteboxplot()
# with a line for the interquartile range and points for whiskers
p + geom_tufteboxplot(median.type = "line", whisker.type = "point", hoffset = 0)
# with a wide line for the interquartile range and lines for whiskers.
# `width` scales the median line but is also the box width, so use
# position = "identity" to stop position_dodge() warning about overlaps.
p + geom_tufteboxplot(median.type = "line", hoffset = 0, width = 3, position = "identity")
# with an offset line for the interquartile range and lines for whiskers
p + geom_tufteboxplot(median.type = "line")
# combined with theme_tufte
p + geom_tufteboxplot() + theme_tufte() + theme(axis.ticks.x = element_blank())
# traditional boxplot with whiskers only out to 1.5 IQR, outlier points
p + geom_tufteboxplot(stat = "boxplot", outlier.shape = 5)
```

---

ggthemes_data *Palette and theme data*

Description

ggthemes_data is a list of the colors, shapes, and other values used in the themes and palettes of this package. Its structure is internal and subject to change.

Usage

```
ggthemes_data
```

Format

A `list` object.

Details

`ggthemes_data$stata$colors$names` spans both generations of Stata's named colors: the classic set plus the `gs0`–`gs16` gray scale, and the `stc1`–`stc15` colors added in Stata 18. See `stata_pal()`.

Colors, shapes and linetypes are stored with the theme they belong to. The Package data article on the package website draws every one of them. `ggthemes_data` contains no fonts: themes set their typefaces in code.

Examples

```
library("scales")

## One element per theme
names(ggthemes_data)

## Colors are stored as tables of names and hex values ...
ggthemes_data$fivethirtyeight
show_col(ggthemes_data$fivethirtyeight$value)

## ... as plain hex vectors ...
show_col(ggthemes_data$manyeyes)

## ... or once per number of colors
show_col(ggthemes_data$solarized$palettes$blue[[4]])

## Shapes give a base R symbol (pch), where there is one, and the Unicode
## character they were transcribed from
stata_shapes <- ggthemes_data$stata$shapes
stata_shapes[c("symbolstyle", "character", "pch")]
show_shapes(stata_shapes$pch)

## Linetypes are dash patterns
show_linetypes(ggthemes_data$stata$linetypes[1:10])
```

---

hc_pal — *Highcharts color palette (discrete)*

Description

Highcharts uses many different color palettes in its plots. This collects the palettes shipped with Highcharts 13, plus the default Highcharts used before v11.

Usage

```
hc_pal(palette = "default")
```

Arguments

palette — A string, the name of the Highcharts palette to use. One of "default", "default_dark", "classic", "darkunica", "grid_light", "sand_signika", "high_contrast_light", "high_contrast_dark", "avocado", "sunset".

Details

"default" and "default_dark" are the light- and dark-mode forms of the palette Highcharts has used by default since v11.0.0; they differ only in positions 2 and 3. "classic" is the default Highcharts used from v5.0.0 through v10.x. The remaining palettes come from the themes bundled with Highcharts.

Note that "avocado" and "sunset" have only four colors.

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

See Also

Other color hc: scale_colour_hc()

Examples

```
hc_pal()(6)

scales::show_col(hc_pal("darkunica")(4))
```

---

numbers_pal — *Apple Numbers color palettes (discrete)*

Description

Color palettes used by charts in Apple Numbers. Each palette provides the six series colors that Numbers assigns to a chart's data series, in order.

Usage

```
numbers_pal(palette = "Classic")
```

Arguments

palette — Palette name. One of "Blue Green", "Blue Violet", "Blue", "Brown", "Classic", "Earth Tone", "Gray", "Green", "Jade", "Mid Century", "Showroom", "Spectrum". The default, "Classic", is the palette Numbers itself uses by default.

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

See Also

Other color numbers: scale_fill_numbers()

Examples

```
library("scales")

show_col(numbers_pal()(6))
show_col(numbers_pal("Spectrum")(6))
```

---

palette_pander *Color palette from the pander package*

Description

The pander package ships with a default colorblind and printer-friendly color palette borrowed from https://jfly.iam.u-tokyo.ac.jp/color/.

Usage

```
palette_pander(n, random_order = FALSE)
```

Arguments

n	Number of colors. This palette supports up to eight colors.
random_order	Whether to shuffle the palette randomly before rendering each plot.

Value

A character vector of n hex colors, recycled if n exceeds the number of colors available. Unlike the other *_pal() functions, this is itself the palette function.

See Also

Other color pander: scale_color_pander()

Examples

```
palette_pander(8)
# the same colors in a random order
palette_pander(8, random_order = TRUE)
```

ptol_pal *Color palettes from Paul Tol's "Colour Schemes"*

Description

[Deprecated]

`ptol_pal()` was deprecated in ggthemes 7.0.0. Use the khroma package instead, which tracks Paul Tol's color schemes as he revises them.

This palette is the 12-color qualitative scheme from Tol's 2012 technical note, and has not followed the revisions he has made since. His current schemes are at https://sronpersonalpages.nl/~pault/; the closest successor to this palette is `khroma::colour("muted")`.

Qualitative color palettes from Paul Tol, "Colour Schemes".

Usage

```
ptol_pal()
```

Details

Incorporation of the palette into an R package was originally inspired by Peter Carl's Paul Tol 21 Gun Salute

Value

A palette function. It takes the number of colors `n` and returns a character vector of `n` hex colors, and can be used as the `palette` argument of `ggplot2::discrete_scale()`.

References

Paul Tol. 2012. "Colour Schemes." SRON Technical Note, SRON/EPS/TN/09-002. https://sronpersonalpages.nl/~pault/data/colourschemes.pdf

See Also

Other color ptol: `scale_colour_ptol()`

Examples

```
library("scales")

show_col(ptol_pal()(6))
show_col(ptol_pal()(4))
show_col(ptol_pal()(12))
```

scale_color_pander *Color scale from the pander package*

Description

The pander package ships with a default colorblind and printer-friendly color palette borrowed from https://jfly.iam.u-tokyo.ac.jp/color/.

Usage

```
scale_color_pander(...)

scale_colour_pander(...)

scale_fill_pander(...)
```

Arguments

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions

- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

`theme_pander()`

Other color pander: `palette_pander()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_pander()
```

`scale_colour_canva` *Discrete color scale using canva.com color palettes*

Description

Color scale for canva.com color palettes described in canva_palettes.

Usage

```
scale_colour_canva(..., palette = "Fresh and bright")

scale_color_canva(..., palette = "Fresh and bright")

scale_fill_canva(..., palette = "Fresh and bright")
```

Arguments

`...`	Arguments passed to `ggplot2::discrete_scale()`.
`palette`	Palette name. See the names of canva_palettes for valid names.

Value

A ggplot2 scale object.

See Also

Other color canva: `canva_pal()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_canva()
```

---

scale_colour_economist

Economist color scales

Description

Color scales using the colors in the Economist graphics.

Usage

```
scale_colour_economist(...)

scale_color_economist(...)

scale_fill_economist(...)
```

Arguments

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

theme_economist() for examples.

Other color economist: economist_pal(), economist_seq_pal(), scale_colour_economist_c()

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_economist()
```

---

scale_colour_economist_c

Economist sequential color scales

Description

Color scales built from the equal-lightness color scales of *The Economist visual styleguide* (v1.2, 4 May 2017); see economist_seq_pal(). The _c scales are continuous; the _ordinal scales are discrete, for ordered factors. See scale_colour_economist() for the unordered categorical scales.

Usage

```
scale_colour_economist_c(hue = "blue", ..., guide = "colourbar")

scale_color_economist_c(hue = "blue", ..., guide = "colourbar")

scale_fill_economist_c(hue = "blue", ..., guide = "colourbar")

scale_colour_economist_ordinal(hue = "blue", ...)

scale_color_economist_ordinal(hue = "blue", ...)

scale_fill_economist_ordinal(hue = "blue", ...)
```

Arguments

hue	A string, the hue of the color scale. One of "blue" (the default), "cyan", "green", "yellow", "olive", "purple", "gold", "gray", or "red".
...	Other arguments passed on to the underlying scale.
guide	A function used to create a guide or its name. See guides() for more information.

Value

A ggplot2 scale object.

See Also

Other color economist: economist_pal(), economist_seq_pal(), scale_colour_economist()

Examples

```
library("ggplot2")

# Continuous scale
ggplot(mtcars, aes(x = wt, y = mpg, colour = disp)) +
  geom_point(size = 3) +
  scale_colour_economist_c(hue = "blue")

# Ordinal (discrete) scale for an ordered factor
ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(cyl, ordered = TRUE))) +
  geom_point(size = 3) +
  scale_colour_economist_ordinal(hue = "red")
```

---

scale_colour_excel_new

Excel (current versions) color scales

Description

Discrete color scales used in current versions of Microsoft Office and Excel.

Usage

```
scale_colour_excel_new(theme = "Office", ...)

scale_color_excel_new(theme = "Office", ...)

scale_fill_excel_new(theme = "Office", ...)
```

Arguments

theme	The name of the Office theme or color theme (not to be confused with ggplot2 themes) from which to derive the color palette. Available themes include: "Atlas", "Badge", "Berlin", "Celestial", "Crop", "Depth", "Droplet", "Facet", "Feathered", "Gallery", "Headlines", "Integral", "Ion Boardroom", "Ion", "Madison", "Main Event", "Mesh", "Office", "Office 2013", "Organic", "Parallax", "Parcel", "Retrospect", "Savon", "Slice", "Vapor Trail", "View", "Wisp", "Wood Type", "Aspect", "Blue Green", "Blue II", "Blue Warm", "Blue", "Grayscale", "Green Yellow", "Green", "Marquee", "Median", "Office 2007", "Orange Red", "Orange", "Paper", "Red Orange", "Red Violet", "Red", "Slipstream", "Violet II", "Violet", "Yellow Orange", "Yellow".

...

Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

Other color excel: excel_new_pal(), excel_pal(), scale_fill_excel()

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_excel_new() + scale_colour_excel_new()
```

---

scale_colour_few *Color scales from Stephen Few*

Description

See few_pal().

Usage

```
scale_colour_few(palette = "Medium", ...)

scale_color_few(palette = "Medium", ...)

scale_fill_few(palette = "Light", ...)
```

Arguments

palette — One of "Light", "Medium", "Dark".

... — Arguments passed on to discrete_scale

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.

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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

Other color few: `few_pal()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_few("Dark")
```

scale_colour_fivethirtyeight

FiveThirtyEight color scales

Description

Color scales using the colors in the FiveThirtyEight graphics.

Usage

```
scale_colour_fivethirtyeight(...)

scale_color_fivethirtyeight(...)

scale_fill_fivethirtyeight(...)
```

Arguments

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions

- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

`theme_fivethirtyeight()` for examples.

Other color fivethirtyeight: `fivethirtyeight_pal()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_fivethirtyeight()
```

`scale_colour_gradient_tableau`

Tableau sequential color scales (continuous)

Description

Continuous color scales using the sequential color palettes in Tableau. See `scale_colour_tableau()` for Tableau discrete color scales, and `scale_colour_gradient2_tableau()` for diverging color scales.

Usage

```
scale_colour_gradient_tableau(
  palette = "Blue",
  ...,
  na.value = "grey50",
  guide = "colourbar"
)

scale_fill_gradient_tableau(
  palette = "Blue",
  ...,
  na.value = "grey50",
  guide = "colourbar"
)

scale_color_gradient_tableau(
  palette = "Blue",
  ...,
  na.value = "grey50",
  guide = "colourbar"
)

scale_color_continuous_tableau(
  palette = "Blue",
  ...,
  na.value = "grey50",
  guide = "colourbar"
)

scale_colour_continuous_tableau(
  palette = "Blue",
  ...,
  na.value = "grey50",
  guide = "colourbar"
)

scale_fill_continuous_tableau(
  palette = "Blue",
  ...,
  na.value = "grey50",
  guide = "colourbar"
)
```

Arguments

`palette` Palette name. For each type:

- `"ordered-sequential"`: `"Blue-Green Sequential"`, `"Blue Light"`, `"Orange Light"`, `"Blue"`, `"Orange"`, `"Green"`, `"Red"`, `"Purple"`, `"Brown"`, `"Gray"`,

"Gray Warm", "Blue-Teal", "Orange-Gold", "Green-Gold", "Red-Gold", "Classic Green", "Classic Gray", "Classic Blue", "Classic Red", "Classic Orange", "Classic Area Red", "Classic Area Green", "Classic Area Brown".

- "ordered-diverging": "Orange-Blue Diverging", "Red-Green Diverging", "Green-Blue Diverging", "Red-Blue Diverging", "Red-Black Diverging", "Gold-Purple Diverging", "Red-Green-Gold Diverging", "Sunset-Sunrise Diverging", "Orange-Blue-White Diverging", "Red-Green-White Diverging", "Green-Blue-White Diverging", "Red-Blue-White Diverging", "Red-Black-White Diverging", "Orange-Blue Light Diverging", "Temperature Diverging", "Classic Red-Green", "Classic Red-Blue", "Classic Red-Black", "Classic Area Red-Green", "Classic Orange-Blue", "Classic Green-Blue", "Classic Red-White-Green", "Classic Red-White-Black", "Classic Orange-White-Blue", "Classic Red-White-Black Light", "Classic Orange-White-Blue Light", "Classic Red-White-Green Light", "Classic Red-Green Light".

...	Arguments passed to tableau_gradient_pal.
na.value	Colour to use for missing values
guide	Type of legend. Use "colourbar" for a continuous color bar, or "legend" for a discrete color legend.

Value

A ggplot2 scale object.

See Also

Other color tableau: scale_colour_gradient2_tableau(), scale_colour_tableau(), tableau_color_pal(), tableau_gradient_pal()

Examples

```
library("ggplot2")

df <- data.frame(
  x = runif(100),
  y = runif(100),
  z1 = rnorm(100),
  z2 = abs(rnorm(100))
)

p <- ggplot(df, aes(x, y)) +
  geom_point(aes(colour = z2)) +
  theme_igray()

palettes <-
  ggthemes_data[["tableau"]][["color-palettes"]][["ordered-sequential"]]
for (palette in head(names(palettes))) {
  print(p + scale_colour_gradient_tableau(palette) + ggtitle(palette))
}
```

scale_colour_gradient2_tableau

Tableau diverging color scales (continuous)

Description

Continuous color scales using the diverging color scales in Tableau. See `scale_colour_tableau()` for Tableau discrete color scales, and `scale_colour_gradient_tableau()` for sequential color scales.

Usage

```
scale_colour_gradient2_tableau(
  palette = "Orange-Blue Diverging",
  ...,
  midpoint = 0,
  na.value = "grey50",
  guide = "colourbar"
)

scale_fill_gradient2_tableau(
  palette = "Orange-Blue Diverging",
  ...,
  midpoint = 0,
  na.value = "grey50",
  guide = "colourbar"
)

scale_color_gradient2_tableau(
  palette = "Orange-Blue Diverging",
  ...,
  midpoint = 0,
  na.value = "grey50",
  guide = "colourbar"
)
```

Arguments

`palette` Palette name. For each type:

- `"ordered-sequential"`: `"Blue-Green Sequential"`, `"Blue Light"`, `"Orange Light"`, `"Blue"`, `"Orange"`, `"Green"`, `"Red"`, `"Purple"`, `"Brown"`, `"Gray"`, `"Gray Warm"`, `"Blue-Teal"`, `"Orange-Gold"`, `"Green-Gold"`, `"Red-Gold"`, `"Classic Green"`, `"Classic Gray"`, `"Classic Blue"`, `"Classic Red"`, `"Classic Orange"`, `"Classic Area Red"`, `"Classic Area Green"`, `"Classic Area Brown"`.

- "ordered-diverging": "Orange-Blue Diverging", "Red-Green Diverging", "Green-Blue Diverging", "Red-Blue Diverging", "Red-Black Diverging", "Gold-Purple Diverging", "Red-Green-Gold Diverging", "Sunset-Sunrise Diverging", "Orange-Blue-White Diverging", "Red-Green-White Diverging", "Green-Blue-White Diverging", "Red-Blue-White Diverging", "Red-Black-White Diverging", "Orange-Blue Light Diverging", "Temperature Diverging", "Classic Red-Green", "Classic Red-Blue", "Classic Red-Black", "Classic Area Red-Green", "Classic Orange-Blue", "Classic Green-Blue", "Classic Red-White-Green", "Classic Red-White-Black", "Classic Orange-White-Blue", "Classic Red-White-Black Light", "Classic Orange-White-Blue Light", "Classic Red-White-Green Light", "Classic Red-Green Light".

...	Arguments passed to tableau_gradient_pal.
midpoint	The data value that corresponds to the middle color of the diverging palette.
na.value	Colour to use for missing values
guide	Type of legend. Use "colourbar" for a continuous color bar, or "legend" for a discrete color legend.

Value

A ggplot2 scale object.

See Also

Other color tableau: scale_colour_gradient_tableau(), scale_colour_tableau(), tableau_color_pal(), tableau_gradient_pal()

Examples

```
library("ggplot2")

df <- data.frame(
  x = runif(100),
  y = runif(100),
  z1 = rnorm(100),
  z2 = abs(rnorm(100))
)
p <- ggplot(df, aes(x, y)) +
  geom_point(aes(colour = z2))

palettes <-
  ggthemes_data[["tableau"]][["color-palettes"]][["ordered-diverging"]]
for (palette in head(names(palettes))) {
  print(p + scale_colour_gradient2_tableau(palette) + ggtitle(palette))
}

# If you need to reverse a palette, use a transformation
p + scale_colour_gradient2_tableau(trans = "reverse")
```

scale_colour_hc *Highcharts color and fill scales*

Description

Color and fill scales which use the palettes in `hc_pal()` and are meant for use with `theme_hc()`.

Usage

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

scale_color_hc(palette = "default", ...)

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

Arguments

`palette` A string, the name of the Highcharts palette to use. One of `"default"`, `"default_dark"`, `"classic"`, `"darkunica"`, `"grid_light"`, `"sand_signika"`, `"high_contrast_light"`, `"high_contrast_dark"`, `"avocado"`, `"sunset"`.

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)

- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

Other color hc: `hc_pal()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_hc(palette = "darkunica")
```

`scale_colour_ptol` *Color scales from Paul Tol's "Colour Schemes"*

Description

[Deprecated]

These scales were deprecated in ggthemes 7.0.0. Use the khroma package instead, which tracks Paul Tol's color schemes as he revises them.

They draw the 12-color qualitative scheme from Tol's 2012 technical note, and have not followed the revisions he has made since. His current schemes are at `https://sronpersonalpages.nl/~pault/`; the closest successor is `khroma::scale_colour_muted()`.

See `ptol_pal()`. These palettes support up to 12 values.

Usage

```
scale_colour_ptol(...)

scale_color_ptol(...)

scale_fill_ptol(...)
```

Arguments

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

Other color ptol: `ptol_pal()`

Examples

```
library("ggplot2")

p2 <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  scale_color_ptol("cyl") +
  theme_minimal() +
  ggtitle("Cars")

ggplot(diamonds, aes(x = clarity, fill = cut)) +
  geom_bar() +
  scale_fill_ptol() +
  theme_minimal()
```

---

scale_colour_stata *Stata color scales*

Description

See `stata_pal()` for details.

Usage

```
scale_colour_stata(scheme = NULL, ...)

scale_fill_stata(scheme = NULL, ...)

scale_color_stata(scheme = NULL, ...)
```

Arguments

`scheme` A string, the Stata scheme. One of: `"s2color"`, `"s1rcolor"`, `"s1color"`, `"mono"`, `"stcolor"`, or `"economist"`. If `NULL`, the default, `"s2color"` is used and a deprecation message is issued; this default becomes `"stcolor"` in ggthemes 8.0.0.

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

Other color stata: `stata_pal()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_stata(scheme = "s2color")
```

---

scale_colour_tableau *Tableau color scales (discrete)*

Description

Categorical (qualitative) color scales used in Tableau. Use the function `scale_colour_gradient_tableau()` for the sequential and `scale_colour_gradient2_tableau()` for the diverging continuous color scales from Tableau.

Usage

```
scale_colour_tableau(
  palette = "Tableau 10",
  ...,
  type = "regular",
  direction = 1
)

scale_fill_tableau(
  palette = "Tableau 10",
  ...,
  type = "regular",
  direction = 1
)

scale_color_tableau(
  palette = "Tableau 10",
  ...,
  type = "regular",
```

```
  direction = 1
)
```

Arguments

palette	Palette name. See tableau_color_pal() for available palettes.
...	Other arguments passed on to ggplot2::discrete_scale().
type	The type of palette, one of: • "regular", the default: discrete, qualitative palettes. • "ordered-sequential": sequential palettes. • "ordered-diverging": diverging palettes.
direction	If 1 (the default), use the original order of colors. If -1, reverse the order.

Value

A ggplot2 scale object.

See Also

tableau_color_pal() for references.

Other color tableau: scale_colour_gradient2_tableau(), scale_colour_gradient_tableau(), tableau_color_pal(), tableau_gradient_pal()

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am) +
  theme_igray()

palettes <- ggthemes_data[["tableau"]][["color-palettes"]][["regular"]]
for (palette in head(names(palettes), 3L)) {
  print(p + scale_colour_tableau(palette) + ggtitle(palette))
}

# the order of colour can be reversed
p + scale_color_tableau(direction = -1)
```

scale_colour_wsj *Wall Street Journal color and fill scales*

Description

Color and fill scales which use the palettes in `wsj_pal()`. These scales should be used with `theme_wsj()`.

Usage

```
scale_colour_wsj(palette = "colors6", ...)

scale_color_wsj(palette = "colors6", ...)

scale_fill_wsj(palette = "colors6", ...)
```

Arguments

`palette` A string, the color palette to use. One of `"rgby"`, `"red_green"`, `"black_green"`, `"dem_rep"`, `"colors6"`.

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)

- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

Other color wsj: `wsj_pal()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_wsj("colors6")
```

`scale_fill_calc` *LibreOffice Calc color scales*

Description

Color scales from LibreOffice Calc.

Usage

```
scale_fill_calc(...)

scale_colour_calc(...)

scale_color_calc(...)
```

Arguments

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

See theme_calc() for examples.

Other color calc: calc_pal()

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_calc()
```

---

scale_fill_excel *Excel 97 ugly color scales*

Description

The classic "ugly" color scales from Excel 97.

Usage

```
scale_fill_excel(...)

scale_colour_excel(...)

scale_color_excel(...)
```

Arguments

`...` Arguments passed on to discrete_scale

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

Other color excel: `excel_new_pal()`, `excel_pal()`, `scale_colour_excel_new()`

Examples

```
library("ggplot2")

# Line and scatter plot colors
p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_excel() + scale_colour_excel()
```

```
# Bar plot (area/fill) colors
ggplot(mpg, aes(x = class, fill = drv)) +
  geom_bar() +
  scale_fill_excel() +
  theme_excel()
```

`scale_fill_gdocs` *Google Docs color scales*

Description

Color scales from Google Docs.

Usage

```
scale_fill_gdocs(...)

scale_colour_gdocs(...)

scale_color_gdocs(...)
```

Arguments

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks

- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

See `theme_gdocs()` for examples.

Other color gdocs: `gdocs_pal()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point(size = 3) +
  scale_colour_gdocs()
```

`scale_fill_numbers` *Apple Numbers color scales*

Description

Discrete color scales using the chart palettes from Apple Numbers.

Usage

```
scale_fill_numbers(palette = "Classic", ...)

scale_colour_numbers(palette = "Classic", ...)

scale_color_numbers(palette = "Classic", ...)
```

Arguments

`palette` Palette name. One of `"Blue Green"`, `"Blue Violet"`, `"Blue"`, `"Brown"`, `"Classic"`, `"Earth Tone"`, `"Gray"`, `"Green"`, `"Jade"`, `"Mid Century"`, `"Showroom"`, `"Spectrum"`. The default, `"Classic"`, is the palette Numbers itself uses by default.

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

super The super class to use for the constructed scale

Value

A ggplot2 scale object.

See Also

See theme_numbers() for examples.

Other color numbers: numbers_pal()

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am) +
  theme_numbers()

for (palette in head(names(ggthemes_data[["numbers"]]), 3L)) {
  print(p + scale_colour_numbers(palette) + ggtitle(palette))
}
```

---

scale_fill_solarized *Solarized color scales*

Description

See solarized_pal() for details.

Usage

```
scale_fill_solarized(accent = "blue", ...)

scale_colour_solarized(accent = "blue", ...)

scale_color_solarized(accent = "blue", ...)
```

Arguments

`accent` A string, the name of the starting color.

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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`.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

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

Value

A ggplot2 scale object.

See Also

Other color solarized: solarized_pal()

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_solarized() + scale_colour_solarized()
```

---

scale_linetype_stata *Stata linetype palette (discrete)*

Description

See stata_linetype_pal() for details.

Usage

```
scale_linetype_stata(...)
```

Arguments

`...` Arguments passed on to discrete_scale

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use show.legend = TRUE.

`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.

`aesthetics` The names of the aesthetics that this scale works with.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

Value

A ggplot2 scale object.

See Also

Other linetype stata: `stata_linetype_pal()`

Examples

```
require("ggplot2")
if (require("tidyr") && require("dplyr")) {
  rescale01 <- function(x) {
    (x - min(x)) / diff(range(x))
  }

  gather(economics, variable, value, -date) |>
    group_by(variable) |>
    mutate(value = rescale01(value)) |>
    ggplot(aes(x = date, y = value, linetype = variable)) +
```

```
    geom_line() +
    scale_linetype_stata()
}
```

---

scale_shape_calc *Calc shape scale*

Description

See calc_shape_pal() for details.

Usage

```
scale_shape_calc(..., unicode = FALSE)
```

Arguments

...

Arguments passed on to discrete_scale

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.

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.

drop Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use show.legend = TRUE.

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.

aesthetics The names of the aesthetics that this scale works with.

minor_breaks One of:

- NULL for no minor breaks
- waiver() for the default breaks (none for discrete, one minor break between each major break for continuous)

- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

`unicode` If `TRUE`, return pch codes derived from Unicode glyphs, as this palette did before ggthemes 6.1.0. Glyph shapes are drawn by the device font, so they render as blank boxes in a font without coverage; the default returns base pch codes, which every font can draw.

Value

A ggplot2 scale object.

See Also

`theme_calc()` for examples.

Other shapes calc: `calc_shape_pal()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, shape = factor(gear))) +
  geom_point(size = 3) +
  scale_shape_calc()
```

`scale_shape_circlefill`

Filled circle shape scale (deprecated)

Description

[Deprecated]

This scale has no direct replacement. For circles that vary by amount of fill, use `scale_shape_tableau(palette = "proportions", unicode = TRUE)`; see `scale_shape_tableau()`.

Usage

```
scale_shape_circlefill(...)
```

Arguments

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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.

`aesthetics` The names of the aesthetics that this scale works with.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

Value

A ggplot2 scale object.

See Also

`circlefill_shape_pal()` for a description of the palette.

Other shapes: `circlefill_shape_pal()`, `cleveland_shape_pal()`, `scale_shape_cleveland()`, `scale_shape_tremmel()`, `tremmel_shape_pal()`

Examples

```
# Deprecated. The palette grades circles by how much of each is filled, which
# needs Unicode glyphs from the device font, so this example does not draw it.
scale_shape_circlefill()
```

`scale_shape_cleveland` *Shape scales from Cleveland "Elements of Graphing Data"*

Description

Shape scales from Cleveland "Elements of Graphing Data"

Usage

```
scale_shape_cleveland(overlap = TRUE, ..., unicode = FALSE)
```

Arguments

`overlap` `logical` Use the scale for overlapping points?

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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.

`aesthetics` The names of the aesthetics that this scale works with.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

`unicode` If `TRUE`, return pch codes derived from Unicode glyphs, as this palette did before ggthemes 6.1.0. Glyph shapes are drawn by the device font, so they render as blank boxes in a font without coverage; the default returns base pch codes, which every font can draw.

Value

A ggplot2 scale object.

References

Cleveland WS. The Elements of Graphing Data. Revised Edition. Hobart Press, Summit, NJ, 1994, pp. 154-164, 234-239.

See Also

cleveland_shape_pal() for a description of the palette.

Other shapes: circlefill_shape_pal(), cleveland_shape_pal(), scale_shape_circlefill(), scale_shape_tremmel(), tremmel_shape_pal()

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, shape = factor(gear))) +
  geom_point(size = 3) +
  scale_shape_cleveland()
```

---

scale_shape_few *Scales for shapes from "Show Me the Numbers"*

Description

scale_shape_few() maps discrete variables to up to five easily discernible shapes. It is based on the shape palette suggested in Few (2012).

Usage

```
scale_shape_few(...)
```

Arguments

...	Common ggplot2::discrete_scale() parameters.

Value

A ggplot2 scale object.

References

Few, S. (2012) *Show Me the Numbers: Designing Tables and Graphs to Enlighten*, Analytics Press, p. 208.

See Also

few_shape_pal() for the shape palette that this scale uses.

Other shapes few: few_shape_pal()

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg, shape = factor(gear))) +
  geom_point(size = 3) +
  scale_shape_few()
```

---

scale_shape_stata *Stata shape scale*

Description

See stata_shape_pal() for details.

Usage

```
scale_shape_stata(..., unicode = FALSE)
```

Arguments

...

Arguments passed on to discrete_scale

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.

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.

drop Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use show.legend = TRUE.

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.

aesthetics The names of the aesthetics that this scale works with.

minor_breaks One of:

- NULL for no minor breaks
- waiver() for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

labels One of the options below. Please note that when labels is a vector, it is highly recommended to also set the breaks argument as a vector to protect against unintended mismatches.

- 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.

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

super The super class to use for the constructed scale

unicode If TRUE, return pch codes derived from Unicode glyphs, as this palette did before ggthemes 6.1.0. Glyph shapes are drawn by the device font, so they render as blank boxes in a font without coverage; the default returns base pch codes, which every font can draw.

Value

A ggplot2 scale object.

See Also

Other shapes stata: stata_shape_pal()

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, shape = factor(gear))) +
  facet_wrap(~am)
p + theme_stata(scheme = "s2color") + scale_shape_stata()
```

scale_shape_tableau *Tableau shape scales*

Description

See tableau_shape_pal() for details.

Usage

```
scale_shape_tableau(palette = "default", ..., unicode = FALSE)
```

Arguments

palette Palette name.

... Arguments passed on to discrete_scale

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.

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.

drop Should unused factor levels be omitted from the scale? The default, TRUE, uses the levels that appear in the data; FALSE includes the levels in the factor. Please note that to display every level in a legend, the layer should use show.legend = TRUE.

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.

aesthetics The names of the aesthetics that this scale works with.

minor_breaks One of:

- NULL for no minor breaks
- waiver() for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions

- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

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

`unicode` If `TRUE`, return pch codes derived from Unicode glyphs, as this palette did before ggthemes 6.1.0. Glyph shapes are drawn by the device font, so they render as blank boxes in a font without coverage; the default returns base pch codes, which every font can draw.

Value

A ggplot2 scale object.

See Also

Other shapes tableau: `tableau_shape_pal()`

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, shape = factor(gear))) +
  facet_wrap(~am)
p + scale_shape_tableau()
```

---

scale_shape_tremmel *Shape scales from Tremmel (1995)*

Description

Shape scales from Tremmel (1995)

Usage

```
scale_shape_tremmel(overlap = FALSE, ..., alt = FALSE)
```

Arguments

`overlap` Use an empty circle instead of a solid circle when `n == 2`.

`...` Arguments passed on to `discrete_scale`

`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.

`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.

`drop` Should unused factor levels be omitted from the scale? The default, `TRUE`, uses the levels that appear in the data; `FALSE` includes the levels in the factor. Please note that to display every level in a legend, the layer should use `show.legend = TRUE`.

`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.

`aesthetics` The names of the aesthetics that this scale works with.

`minor_breaks` One of:

- `NULL` for no minor breaks
- `waiver()` for the default breaks (none for discrete, one minor break between each major break for continuous)
- A numeric vector of positions
- A function that given the limits returns a vector of minor breaks. Also accepts rlang lambda function notation. When the function has two arguments, it will be given the limits and major break positions.

`labels` One of the options below. Please note that when `labels` is a vector, it is highly recommended to also set the `breaks` argument as a vector to protect against unintended mismatches.

- `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.

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

super The super class to use for the constructed scale

alt If TRUE, then when n == 3, use a solid circle, plus sign and empty triangle. Otherwise use a solid circle, empty circle, and empty triangle. Defaults to FALSE, the triple Tremmel's Experiment 1 actually measured; the TRUE triple is argued on feature-dimension grounds that Tremmel flags as not directly supported by the experiments.

Value

A ggplot2 scale object.

See Also

tremmel_shape_pal() for a description of the palette.

Other shapes: circlefill_shape_pal(), cleveland_shape_pal(), scale_shape_circlefill(), scale_shape_cleveland(), tremmel_shape_pal()

Examples

```
library("ggplot2")

p <- ggplot(mtcars, aes(x = mpg, y = hp, shape = factor(cyl))) +
  geom_point()

p + scale_shape_tremmel()
p + scale_shape_tremmel(alt = TRUE)
p + scale_shape_tremmel(overlap = TRUE)
```

---

show_linetypes *Show linetypes*

Description

A quick and dirty way to show linetypes.

Usage

```
show_linetypes(linetypes, labels = TRUE)
```

Arguments

linetypes A character vector of linetypes. See graphics::par().

labels If TRUE (the default), label each line with its linetype (lty) value.

Value

Called for its side effect of creating a plot; returns linetypes invisibly.

See Also

scales::show_col(), show_shapes()

Examples

```
library("scales")

show_linetypes(linetype_pal()(3))
show_linetypes(linetype_pal()(3), labels = TRUE)
```

---

show_shapes *Show shapes*

Description

A quick and dirty way to show shapes.

Usage

```
show_shapes(shapes, labels = TRUE)
```

Arguments

shapes	A numeric or character vector of shapes. See graphics::par().
labels	If TRUE (the default), label each symbol with its plotting character value.

Value

Called for its side effect of creating a plot; returns shapes invisibly.

See Also

scales::show_col(), show_linetypes()

Examples

```
library("scales")

show_shapes(shape_pal()(5))
show_shapes(shape_pal()(3), labels = TRUE)
```

smart_digits *Format numbers with automatic number of digits*

Description

Format numbers with automatic number of digits

Usage

```
smart_digits(x, ...)

smart_digits_format(x, ...)
```

Arguments

x	A numeric vector to format.
...	Parameters passed to `base::format()`.

Value

A character vector. `smart_digits_format()` returns a function with a single argument `x`, a numeric vector, that returns a character vector.

Author(s)

Josh O'Brien, Baptiste Auguie, Jeffrey B. Arnold

References

Josh O'Brien, https://stackoverflow.com/questions/23169938/select-accuracy-to-display-additional-axis-
23171858#23171858.

Examples

```
smart_digits(c(0.1234, 0.5, 1.25))

smart_digits(c(1234.5678, 2000, 10000))

# A labelling function for use in a scale
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  scale_y_continuous(labels = smart_digits_format())
```

solarized_pal *Solarized color palette (discrete)*

Description

Qualitative color palette based on Ethan Schoonover's Solarized palette, https://ethanschoonover.com/solarized/. This palette supports up to seven values.

Usage

```
solarized_pal(accent = "blue")
```

Arguments

accent	A string, the name of the starting color.

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

Note

For a given starting color and number of colors in the palette, the other colors are the combination of colors that maximizes the total Euclidean distance between colors in L*a*b space.

See Also

Other color solarized: scale_fill_solarized()

Examples

```
library("scales")

show_col(solarized_pal()(2))
show_col(solarized_pal()(3))
show_col(solarized_pal("red")(4))
```

stat_fivenumber *Calculate components of a five-number summary*

Description

The five number summary of a sample is the minimum, first quartile, median, third quartile, and maximum.

Usage

```
stat_fivenumber(
  mapping = NULL,
  data = NULL,
  geom = "boxplot",
  probs = c(0, 0.25, 0.5, 0.75, 1),
  na.rm = FALSE,
  position = "identity",
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

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

`data` The data to be displayed in this layer. There are three options:

If `NULL`, the default, the data is inherited from the plot data as specified in the call to `ggplot()`.

A `data.frame`, or other object, will override the plot data. All objects will be fortified to produce a data frame. See `fortify()` for which variables will be created.

A `function` will be called with a single argument, the plot data. The return value must be a `data.frame`, and will be used as the layer data. A `function` can be created from a `formula` (e.g. `~ head(.x, 10)`).

`geom` The geometric object to use to display the data for this layer. When using a `stat_*()` function to construct a layer, the `geom` argument can be used to override the default coupling between stats and geoms. The `geom` argument accepts the following:

- A `Geom` ggproto subclass, for example `GeomPoint`.
- A string naming the geom. To give the geom as a string, strip the function name of the `geom_` prefix. For example, to use `geom_point()`, give the geom as `"point"`.
- For more information and other ways to specify the geom, see the layer geom documentation.

`probs` Quantiles to use for the five number summary.

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

`position` A position adjustment to use on the data for this layer. This can be used in various ways, including to prevent overplotting and improving the display. The `position` argument accepts the following:

- The result of calling a position function, such as `position_jitter()`. This method allows for passing extra arguments to the position.
- A string naming the position adjustment. To give the position as a string, strip the function name of the `position_` prefix. For example, to use `position_jitter()`, give the position as `"jitter"`.
- For more information and other ways to specify the position, see the layer position documentation.

`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. To include legend keys for all levels, even when no data exists, use `TRUE`. If `NA`, all levels are shown in legend, but unobserved levels are omitted.

`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. `annotation_borders()`.

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

- Static aesthetics that are not mapped to a scale, but are at a fixed value and apply to the layer as a whole. For example, `colour = "red"` or `linewidth = 3`. The geom's documentation has an **Aesthetics** section that lists the available options. The 'required' aesthetics cannot be passed on to the `params`. Please note that while passing unmapped aesthetics as vectors is technically possible, the order and required length is not guaranteed to be parallel to the input data.
- When constructing a layer using a `stat_*()` function, the `...` argument can be used to pass on parameters to the `geom` part of the layer. An example of this is `stat_density(geom = "area", outline.type = "both")`. The geom's documentation lists which parameters it can accept.
- Inversely, when constructing a layer using a `geom_*()` function, the `...` argument can be used to pass on parameters to the `stat` part of the layer. An example of this is `geom_area(stat = "density", adjust = 0.5)`. The stat's documentation lists which parameters it can accept.
- The `key_glyph` argument of `layer()` may also be passed on through `...`. This can be one of the functions described as key glyphs, to change the display of the layer in the legend.

Value

A data frame with additional columns:

- `width`: width of the box plot.
- `min`: minimum.
- `lower`: lower hinge, the 25% quantile.
- `middle`: median, the 50% quantile.
- `upper`: upper hinge, the 75% quantile.
- `max`: maximum.

See Also

`ggplot2::stat_boxplot()`

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  stat_fivenumber()

# The whiskers can be set to other quantiles
ggplot(mtcars, aes(x = factor(cyl), y = mpg)) +
  stat_fivenumber(probs = c(0.05, 0.25, 0.5, 0.75, 0.95))
```

---

stata_linetype_pal *Stata linetype palette (discrete)*

Description

Linetype palette based on the linepattern scheme in Stata. This palette supports up to 15 values.

Usage

```
stata_linetype_pal()
```

Value

A palette function. It takes the number of linetypes `n` and returns a character vector of `n` linetype specifications, and can be used as the `palette` argument of `ggplot2::discrete_scale()`.

See Also

`scale_linetype_stata()`

Other linetype stata: `scale_linetype_stata()`

Examples

```
stata_linetype_pal()(6)

show_linetypes(stata_linetype_pal()(6))
```

stata_pal *Stata color palettes (discrete)*

Description

Stata color palettes. See Stata documentation for a description of the schemes, https://www.stata.com/help.cgi?schemes.

Usage

```
stata_pal(scheme = NULL)
```

Arguments

scheme	A string, the Stata scheme. One of: "s2color", "s1rcolor", "s1color", "mono", "stcolor", or "economist". If NULL, the default, "s2color" is used and a deprecation message is issued; this default becomes "stcolor" in ggthemes 8.0.0.

Details

All these palettes support up to 15 values.

Stata's palettes come in two generations, and both are included here.

- **Stata 17 and earlier:** schemes "s2color", "s1color", "s1rcolor", and "mono" are built from Stata's classic named colors (navy, maroon, forest_green, and so on) and the gs0–gs16 gray scale. "s2color" was Stata's factory default through Stata 17.
- **Stata 18 and later:** scheme "stcolor" uses the stc1–stc15 colors introduced in Stata 18. They are brighter than the classic palette and chosen to stay distinguishable for readers with a color vision deficiency. The first four are also available under the aliases stblue, stred, stgreen, and styellow. "stcolor" has been Stata's factory default since Stata 18.

"economist" is not one of Stata's general-purpose schemes; it is the set of Economist-styled colors that Stata ships in scheme-economist.scheme.

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

See Also

Other color stata: scale_colour_stata()

Examples

```
library("scales")

# Stata 18 and later (the current factory default)
show_col(stata_pal("stcolor")(15))

# Stata 17 and earlier
show_col(stata_pal("s2color")(15))
show_col(stata_pal("s1rcolor")(15))
show_col(stata_pal("s1color")(15))
show_col(stata_pal("mono")(15))
```

---

stata_shape_pal *Stata shape palette (discrete)*

Description

Shape palette based on the symbol palette in Stata used in scheme s2mono. This palette supports up to 10 values.

Usage

```
stata_shape_pal(unicode = FALSE)
```

Arguments

unicode	If TRUE, return pch codes derived from Unicode glyphs, as this palette did before ggthemes 6.1.0. Glyph shapes are drawn by the device font, so they render as blank boxes in a font without coverage; the default returns base pch codes, which every font can draw.

Value

A palette function. It takes the number of shapes n and returns an integer vector of n shape (pch) codes, and can be used as the palette argument of ggplot2::discrete_scale().

Note

Stata's ten plotting symbols all have a base pch equivalent, so this palette supports ten values on either branch and nothing is dropped: solid and hollow circle, diamond, square and triangle, plus the X and the plus sign.

See Also

See scale_shape_stata() for examples.

Other shapes stata: scale_shape_stata()

Examples

```
stata_shape_pal()(10)

show_shapes(stata_shape_pal()(10))
```

`tableau_color_pal` *Tableau color palettes (discrete)*

Description

Color palettes used in Tableau.

Usage

```
tableau_color_pal(
  palette = "Tableau 10",
  type = c("regular", "ordered-sequential", "ordered-diverging"),
  direction = 1
)

tableau_colour_pal(
  palette = "Tableau 10",
  type = c("regular", "ordered-sequential", "ordered-diverging"),
  direction = 1
)
```

Arguments

`palette`	A string, the palette name. See Details for available palettes.
`type`	The type of palette, one of: • `"regular"`, the default: discrete, qualitative palettes. • `"ordered-sequential"`: sequential palettes. • `"ordered-diverging"`: diverging palettes.
`direction`	If `1` (the default), use the original order of colors. If `-1`, reverse the order.

Details

Tableau provides three types of color palettes: `"regular"` (discrete, qualitative categories), `"ordered-sequential"`, and `"ordered-diverging"`.

- `"regular"`: `"Tableau 10"`, `"Tableau 20"`, `"Color Blind"`, `"Seattle Grays"`, `"Traffic"`, `"Miller Stone"`, `"Superfishel Stone"`, `"Nuriel Stone"`, `"Jewel Bright"`, `"Summer"`, `"Winter"`, `"Green-Orange-Teal"`, `"Blue-Red-Brown"`, `"Purple-Pink-Gray"`, `"Hue Circle"`, `"Classic 10"`, `"Classic 10 Medium"`, `"Classic 10 Light"`, `"Classic 20"`, `"Classic Gray 5"`, `"Classic Color Blind"`, `"Classic Traffic Light"`, `"Classic Purple-Gray 6"`, `"Classic Purple-Gray 12"`, `"Classic Green-Orange 6"`, `"Classic Green-Orange 12"`, `"Classic Blue-Red 6"`, `"Classic Blue-Red 12"`, `"Classic Cyclic"`.

- "ordered-diverging": "Orange-Blue Diverging", "Red-Green Diverging", "Green-Blue Diverging", "Red-Blue Diverging", "Red-Black Diverging", "Gold-Purple Diverging", "Red-Green-Gold Diverging", "Sunset-Sunrise Diverging", "Orange-Blue-White Diverging", "Red-Green-White Diverging", "Green-Blue-White Diverging", "Red-Blue-White Diverging", "Red-Black-White Diverging", "Orange-Blue Light Diverging", "Temperature Diverging", "Classic Red-Green", "Classic Red-Blue", "Classic Red-Black", "Classic Area Red-Green", "Classic Orange-Blue", "Classic Green-Blue", "Classic Red-White-Green", "Classic Red-White-Black", "Classic Orange-White-Blue", "Classic Red-White-Black Light", "Classic Orange-White-Blue Light", "Classic Red-White-Green Light", "Classic Red-Green Light".
- "ordered-sequential": "Blue-Green Sequential", "Blue Light", "Orange Light", "Blue", "Orange", "Green", "Red", "Purple", "Brown", "Gray", "Gray Warm", "Blue-Teal", "Orange-Gold", "Green-Gold", "Red-Gold", "Classic Green", "Classic Gray", "Classic Blue", "Classic Red", "Classic Orange", "Classic Area Red", "Classic Area Green", "Classic Area Brown".

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

References

http://vis.stanford.edu/color-names/analyzer/

Maureen Stone, 'Designing Colors for Data' (slides), at the International Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging, Banff, AB, Canada, June 22, 2007.

Heer, Jeffrey and Maureen Stone, 2012 'Color Naming Models for Color Selection, Image Editing and Palette Design', ACM Human Factors in Computing Systems (CHI) http://vis.stanford.edu/files/2012-ColorNameModels-CHI.pdf.

See Also

Other color tableau: scale_colour_gradient2_tableau(), scale_colour_gradient_tableau(), scale_colour_tableau(), tableau_gradient_pal()

Examples

```
library("scales")

palettes <- ggthemes_data[["tableau"]][["color-palettes"]][["regular"]]
for (palname in names(palettes)) {
  pal <- tableau_color_pal(palname)
  max_n <- attr(pal, "max_n")
  show_col(pal(max_n))
  title(main = palname)
}
```

tableau_gradient_pal *Tableau color gradient palettes (continuous)*

Description

Gradient color palettes using the diverging and sequential continuous color palettes in Tableau. See `tableau_color_pal()` for discrete color palettes.

Usage

```
tableau_gradient_pal(palette = "Blue", type = "ordered-sequential")

tableau_seq_gradient_pal(palette = "Blue", ...)

tableau_div_gradient_pal(palette = "Orange-Blue Diverging", ...)
```

Arguments

`palette` Palette name. For each `type`:

- `"ordered-sequential"`: `"Blue-Green Sequential"`, `"Blue Light"`, `"Orange Light"`, `"Blue"`, `"Orange"`, `"Green"`, `"Red"`, `"Purple"`, `"Brown"`, `"Gray"`, `"Gray Warm"`, `"Blue-Teal"`, `"Orange-Gold"`, `"Green-Gold"`, `"Red-Gold"`, `"Classic Green"`, `"Classic Gray"`, `"Classic Blue"`, `"Classic Red"`, `"Classic Orange"`, `"Classic Area Red"`, `"Classic Area Green"`, `"Classic Area Brown"`.
- `"ordered-diverging"`: `"Orange-Blue Diverging"`, `"Red-Green Diverging"`, `"Green-Blue Diverging"`, `"Red-Blue Diverging"`, `"Red-Black Diverging"`, `"Gold-Purple Diverging"`, `"Red-Green-Gold Diverging"`, `"Sunset-Sunrise Diverging"`, `"Orange-Blue-White Diverging"`, `"Red-Green-White Diverging"`, `"Green-Blue-White Diverging"`, `"Red-Blue-White Diverging"`, `"Red-Black-White Diverging"`, `"Orange-Blue Light Diverging"`, `"Temperature Diverging"`, `"Classic Red-Green"`, `"Classic Red-Blue"`, `"Classic Red-Black"`, `"Classic Area Red-Green"`, `"Classic Orange-Blue"`, `"Classic Green-Blue"`, `"Classic Red-White-Green"`, `"Classic Red-White-Black"`, `"Classic Orange-White-Blue"`, `"Classic Red-White-Black Light"`, `"Classic Orange-White-Blue Light"`, `"Classic Red-White-Green Light"`, `"Classic Red-Green Light"`.

`type` The type of palette, one of:

- `"ordered-sequential"`, the default: sequential palettes.
- `"ordered-diverging"`: diverging palettes.

`...` Arguments passed to `tableau_gradient_pal`.

Value

A palette function. It takes a numeric vector `x` of values between 0 and 1 and returns a character vector of hex colors interpolated along the palette, for use as the `palette` argument of `ggplot2::continuous_scale()`.

See Also

Other color tableau: scale_colour_gradient2_tableau(), scale_colour_gradient_tableau(), scale_colour_tableau(), tableau_color_pal()

Examples

```
library("scales")

x <- seq(0, 1, length = 25)
r <- sqrt(outer(x^2, x^2, "+"))
palettes <-
  ggthemes_data[["tableau"]][["color-palettes"]][["ordered-sequential"]]
for (palname in names(palettes)) {
  col <- tableau_seq_gradient_pal(palname)(seq(0, 1, length = 12))
  image(r, col = col)
  title(main = palname)
}
```

---

tableau_shape_pal *Tableau shape palettes (discrete)*

Description

Shape palettes used by Tableau.

Usage

```
tableau_shape_pal(
  palette = c("default", "filled", "proportions"),
  unicode = FALSE
)
```

Arguments

palette	Palette name.
unicode	If TRUE, return pch codes derived from Unicode glyphs, as this palette did before ggthemes 6.1.0. Glyph shapes are drawn by the device font, so they render as blank boxes in a font without coverage; the default returns base pch codes, which every font can draw.

Details

Not all shape palettes in Tableau are supported, and these palettes are not exact.

Shape palettes in Tableau are used to expose images for use as markers in charts, and thus are sometimes groupings of closely related symbols.

Value

A palette function. It takes the number of shapes `n` and returns an integer vector of `n` shape (`pch`) codes, and can be used as the `palette` argument of `ggplot2::discrete_scale()`.

Note

Supported values by palette: `"default"` eight (ten with `unicode = TRUE`), `"filled"` six (ten), `"proportions"` two (five). Shapes with no base pch equivalent – the sideways triangles, the solid star, and the partially filled circles – are dropped rather than approximated by a different shape.

`"proportions"` encodes *fill fraction*, which base pch cannot express at all, so only its empty and full circles survive; they remain meaningful as a two-value scale. To encode a proportion, map `alpha` or `fill` instead, or use `unicode = TRUE` with a font covering Geometric Shapes, such as DejaVu Sans.

See Also

Other shapes tableau: `scale_shape_tableau()`

Examples

```
# The eight shapes with a font-independent equivalent.
show_shapes(tableau_shape_pal()(8))

## Not run:
  # All ten, drawn from the device font. Needs a font covering Geometric
  # Shapes, such as DejaVu Sans.
  show_shapes(tableau_shape_pal(unicode = TRUE)(10))

## End(Not run)
```

---

theme_base *Base R graphics theme*

Description

Theme similar to the default settings of the "base" R graphics.

Usage

```
theme_base(base_size = 16, base_family = "")
```

Arguments

`base_size`	base font size, given in pts.
`base_family`	base font family

Value

A ggplot2 theme object (class theme).

See Also

Other themes: theme_clean(), theme_foundation(), theme_igray(), theme_map(), theme_par(), theme_solid()

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(
    x = wt,
    y = mpg,
    colour = factor(gear)
  )) +
  facet_wrap(~am)
p + theme_base()
# Change values of par
old_par <- par(fg = "blue", bg = "gray", col.lab = "red", font.lab = 3)
p + theme_base()
par(old_par)
```

---

theme_calc *LibreOffice Calc theme*

Description

Theme similar to the default settings of LibreOffice Calc charts.

Usage

```
theme_calc(base_size = 10, base_family = "sans")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

Value

A ggplot2 theme object (class theme).

Examples

```
library("ggplot2")

ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am) +
  theme_calc() +
  scale_color_calc()

ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, shape = factor(gear))) +
  facet_wrap(~am) +
  theme_calc() +
  scale_shape_calc()
```

---

theme_clean *Clean ggplot theme*

Description

Clean ggplot theme with no panel background, black axis lines and gray fill color for chart elements.

Usage

```
theme_clean(base_size = 12, base_family = "sans")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

Value

A ggplot2 theme object (class theme).

Author(s)

Konrad Zdeb name.surname@me.com

See Also

Other themes: theme_base(), theme_foundation(), theme_igray(), theme_map(), theme_par(), theme_solid()

Examples

```
library("ggplot2")
p <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point() +
  facet_wrap(~am) +
  geom_smooth(method = "lm", se = FALSE) +
  theme_clean()
p
```

---

theme_economist *ggplot color theme based on the Economist*

Description

A theme that approximates the style of *The Economist*.

Usage

```
theme_economist(
  base_size = 10,
  base_family = "sans",
  horizontal = TRUE,
  dkpanel = FALSE
)

theme_economist_white(
  base_size = 11,
  base_family = "sans",
  gray_bg = TRUE,
  horizontal = TRUE
)
```

Arguments

base_size	base font size, given in pts.
base_family	base font family
horizontal	If TRUE (the default), draw horizontal grid lines.
dkpanel	If TRUE, use a darker background for the panel region. The default is FALSE.
gray_bg	If TRUE (the default), use a gray background; otherwise, use a white background.

Details

theme_economist() implements the standard bluish-gray background theme in the print *The Economist* and economist.com.

theme_economist_white() implements a variant with a white panel and light gray (or white) background often used by *The Economist* blog Graphic Detail.

Use `scale_color_economist()` with this theme. The y axis should be displayed on the right hand side.

The Economist uses "ITC Officina Sans" as its font for graphs. If you have access to this font, you can use it with the extrafont package. "Verdana" is a good substitute.

Value

An object of class `ggplot2::theme()`.

References

- The Economist
- Spiekerblog, "ITC Officina Display", January 1, 2007.

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
     geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
     facet_wrap(~am) +
     # Economist puts x-axis labels on the right-hand side
     scale_y_continuous(position = "right")

## Standard
p + theme_economist() +
  scale_colour_economist()

# Change axis lines to vertical
p + theme_economist(horizontal = FALSE) +
    scale_colour_economist() +
    coord_flip()

## White panel/light gray background
p + theme_economist_white() +
    scale_colour_economist()

## All white variant
p + theme_economist_white(gray_bg = FALSE) +
    scale_colour_economist()

## Not run:

## The Economist uses ITC Officina Sans
library("extrafont")
p + theme_economist(base_family="ITC Officina Sans") +
    scale_colour_economist()

## Verdana is a widely available substitute
p + theme_economist(base_family="Verdana") +
    scale_colour_economist()
```

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

---

theme_excel *ggplot theme based on old Excel plots*

Description

Theme to replicate the ugly monstrosity that was the old gray-background Excel chart. Please never use this. This theme should be combined with the `scale_colour_excel()` color scale.

Usage

```
theme_excel(base_size = 12, base_family = "", horizontal = TRUE)
```

Arguments

`base_size`	base font size, given in pts.
`base_family`	base font family
`horizontal`	If `TRUE` (the default), draw horizontal grid lines.

Value

An object of class `ggplot2::theme()`.

See Also

Other themes excel: `theme_excel_new()`

Examples

```
library("ggplot2")

# Line and scatter plot colors
p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_excel() + scale_colour_excel()

# Bar plot (area/fill) colors
ggplot(mpg, aes(x = class, fill = drv)) +
  geom_bar() +
  scale_fill_excel() +
  theme_excel()
```

theme_excel_new *ggplot theme similar to current Excel plot defaults*

Description

Theme for ggplot2 that is similar to the default style of charts in current versions of Microsoft Excel.

Usage

```
theme_excel_new(base_size = 9, base_family = "sans")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

Details

Excel derives its chart grays from the theme's `tx1` color by luminance transform rather than hard-coding them. Since `tx1` is black in every built-in Office theme, these grays—`"#D9D9D9"` gridlines, `"#BFBFBF"` axis lines, `"#595959"` text—are the same whichever theme `scale_colour_excel_new()` is set to.

Since 2023 the default font in Excel has been Aptos, but `base_family` defaults to `"sans"` because Aptos is rarely installed outside of Office. Pass `base_family = "Aptos Narrow"` for a closer match if you do have it.

Value

An object of class `ggplot2::theme()`.

See Also

Other themes excel: `theme_excel()`

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_excel_new() + scale_colour_excel_new()
```

theme_few *Theme based on Few's "Practical Rules for Using Color in Charts"*

Description

Theme based on the rules and examples from Stephen Few's *Show Me the Numbers* and "Practical Rules for Using Color in Charts".

Usage

```
theme_few(base_size = 12, base_family = "")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

Value

A ggplot2 theme object (class theme).

References

Few, S. (2012) *Show Me the Numbers: Designing Tables and Graphs to Enlighten*. 2nd edition. Analytics Press.

Stephen Few, "Practical Rules for Using Color in Charts", https://www.perceptualedge.com/articles/visual_business_intelligence/rules_for_using_color.pdf.

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_few() + scale_colour_few()
p + theme_few() + scale_colour_few("Light")
p + theme_few() + scale_colour_few("Dark")

ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, shape = factor(gear))) +
  theme_few() +
  scale_shape_few()
```

theme_fivethirtyeight *Theme inspired by FiveThirtyEight plots*

Description

Theme inspired by the plots from FiveThirtyEight.com.

Usage

```
theme_fivethirtyeight(base_size = 12, base_family = "sans")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

Value

A ggplot2 theme object (class theme).

Examples

```
library("ggplot2")
p <- ggplot(mtcars, aes(x = wt, y = mpg, colour = factor(gear))) +
  geom_point() +
  facet_wrap(~am) +
  geom_smooth(method = "lm", se = FALSE) +
  scale_color_fivethirtyeight() +
  theme_fivethirtyeight()
p
```

theme_foundation *Foundation theme*

Description

This theme is designed to be a foundation from which to build new themes, and not meant to be used directly. theme_foundation() is a complete theme with only minimal number of elements defined. It is easier to create new themes by extending this one rather than ggplot2::theme_gray() or ggplot2::theme_bw(), because those themes define elements deep in the hierarchy.

Usage

```
theme_foundation(
  base_size = 12,
  base_family = "",
  ink = "black",
  paper = "white",
  accent = "#3366FF"
)
```

Arguments

base_size	base font size, given in pts.
base_family	base font family
ink, paper, accent	colour for foreground, background, and accented elements respectively.

Details

This theme takes ggplot2::theme_gray() and sets all colour and fill values to NULL, except for the top-level elements (line, rect, and title), which have colour = ink, and fill = paper. This leaves the spacing and non-color defaults of the default ggplot2 themes in place.

Unlike theme_foundation(), the other themes in this package (e.g. theme_economist(), theme_excel(), theme_hc()) intentionally replicate a fixed, published visual style, so they do not expose ink/paper/accent arguments.

Value

A ggplot2 theme object (class theme).

See Also

Other themes: theme_base(), theme_clean(), theme_igray(), theme_map(), theme_par(), theme_solid()

Examples

```
library("ggplot2")

ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  theme_foundation()

# Extend the foundation to build a new theme
theme_minimal_box <- function(...) {
  theme_foundation(...) +
    theme(
      panel.grid = element_blank(),
      axis.ticks = element_line()
    )
```

```
}

ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  theme_minimal_box()
```

---

theme_gdocs *Theme with Google Docs chart defaults*

Description

Theme similar to the default look of charts in Google Docs.

Usage

```
theme_gdocs(base_size = 12, base_family = "sans")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

Value

A ggplot2 theme object (class theme).

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_gdocs() + scale_color_gdocs()
```

---

theme_hc *Highcharts theme*

Description

Themes based on Highcharts plots.

Usage

```
theme_hc(
  base_size = 12,
  base_family = "sans",
 style = c("default", "default_dark", "darkunica", "grid_light", "sand_signika"),
  bgcolor = lifecycle::deprecated()
)
```

Arguments

base_size	base font size, given in pts.
base_family	base font family
style	The Highcharts theme to use. One of "default", "default_dark", "darkunica", "grid_light", "sand_signika".
bgcolor	**[Deprecated]** Use style instead.

Details

Only the Highcharts themes that restyle the chart itself get a style here. The "high-contrast", "avocado" and "sunset" themes shipped with Highcharts 13 change nothing but the series colors, so they are available through hc_pal() alone; combine them with theme_hc("default") or theme_hc("default_dark").

Highcharts pairs several of these themes with a web font (darkunica with Unica One, grid_light with Dosis, sand_signika with Signika). Those are not requested here, since the font may not be installed; pass base_family to use one.

Value

A ggplot2 theme object (class theme).

References

https://www.highcharts.com/demo/highcharts/line-chart

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(
    x = wt,
    y = mpg,
    colour = factor(gear)
  )) +
  facet_wrap(~am)
p + theme_hc() + scale_colour_hc()
p + theme_hc(style = "darkunica") + scale_colour_hc("darkunica")
p + theme_hc(style = "grid_light") + scale_colour_hc("grid_light")
p + theme_hc(style = "default_dark") + scale_colour_hc("high_contrast_dark")
```

```
dtemp <- data.frame(
  months = factor(rep(substr(month.name, 1, 3), 4), levels = substr(month.name, 1, 3)),
  city = rep(c("Tokyo", "New York", "Berlin", "London"), each = 12),
  temp = c(
    7.0,
    6.9,
    9.5,
    14.5,
    18.2,
    21.5,
    25.2,
    26.5,
    23.3,
    18.3,
    13.9,
    9.6,
    -0.2,
    0.8,
    5.7,
    11.3,
    17.0,
    22.0,
    24.8,
    24.1,
    20.1,
    14.1,
    8.6,
    2.5,
    -0.9,
    0.6,
    3.5,
    8.4,
    13.5,
    17.0,
    18.6,
    17.9,
    14.3,
    9.0,
    3.9,
    1.0,
    3.9,
    4.2,
    5.7,
    8.5,
    11.9,
    15.2,
    17.0,
    16.6,
    14.2,
    10.3,
    6.6,
    4.8
```

```
  )
)

ggplot(dtemp, aes(x = months, y = temp, group = city, color = city)) +
  geom_line() +
  geom_point(size = 1.1) +
  ggtitle("Monthly Average Temperature") +
  theme_hc() +
  scale_colour_hc()

ggplot(dtemp, aes(x = months, y = temp, group = city, color = city)) +
  geom_line() +
  geom_point(size = 1.1) +
  ggtitle("Monthly Average Temperature") +
  theme_hc(style = "darkunica") +
  scale_colour_hc("darkunica")
```

`theme_igray` *Inverse gray theme*

Description

Theme with white panel and gray background.

Usage

```
theme_igray(base_size = 12, base_family = "")
```

Arguments

`base_size`	base font size, given in pts.
`base_family`	base font family

Details

This theme inverts the colors of `ggplot2::theme_gray()`, with a white panel and a light gray area around it. This keeps a white background for the color scales, like `ggplot2::theme_bw()`. But by using a gray background, the plot is closer to the typographical color of the document, which is the motivation for using a gray panel in `ggplot2::theme_gray()`. This is similar to the style of plots in Stata and Tableau.

Value

A ggplot2 theme object (class `theme`).

See Also

`ggplot2::theme_gray()`, `ggplot2::theme_bw()`

Other themes: `theme_base()`, `theme_clean()`, `theme_foundation()`, `theme_map()`, `theme_par()`, `theme_solid()`

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_igray()
```

---

theme_map *Clean theme for maps*

Description

A clean theme that is good for displaying maps from ggplot2::geom_map().

Usage

```
theme_map(base_size = 9, base_family = "")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

Value

A ggplot2 theme object (class theme).

See Also

Other themes: theme_base(), theme_clean(), theme_foundation(), theme_igray(), theme_par(), theme_solid()

Examples

```
library("ggplot2")

if (requireNamespace("maps", quietly = TRUE) && requireNamespace("mapproj", quietly = TRUE)) {
  us <- map_data("state")
  gg <- ggplot(us, aes(x = long, y = lat, group = group)) +
    geom_polygon(fill = "white", color = "black", linewidth = 0.25) +
    coord_map("albers", lat0 = 39, lat1 = 45) +
    theme_map()
  gg
}
```

theme_numbers *Theme with Apple Numbers chart defaults*

Description

Theme similar to the default look of charts in Apple Numbers.

Usage

```
theme_numbers(base_size = 12, base_family = "sans")
```

Arguments

base_size	base font size, given in pts.
base_family	base font family

Details

The values used here are those in the `chart-style-default` style of the theme stylesheet that ships inside Numbers: no chart background fill, gridlines in the value direction only, a border along the bottom of the chart but not the other three sides, and no tick marks.

Value

A ggplot2 theme object (class `theme`).

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)
p + theme_numbers() + scale_color_numbers()
```

theme_pander *A ggplot theme originated from the pander package*

Description

The pander package ships with a default theme when the 'unify plots' option is enabled via `panderOptions`, which is now also available outside of pander internals, like `evals`, `eval.msgs` or `Pandoc.brew`.

Usage

```
theme_pander(
  base_size = 12,
  base_family = "sans",
  nomargin = TRUE,
  ff = NULL,
  fc = "black",
  fs = NULL,
  gM = TRUE,
  gm = TRUE,
  gc = "grey",
  gl = "dashed",
  boxes = FALSE,
  bc = "white",
  pc = "transparent",
  lp = "right",
  axis = 1
)
```

Arguments

`base_size`	base font size, given in pts.
`base_family`	base font family
`nomargin`	Whether to suppress the white space around the plot.
`ff`	Font family, like `"sans"`. **[Deprecated]** Use `base_family` instead.
`fc`	Font color, as a name or hex code.
`fs`	Font size (integer). **[Deprecated]** Use `base_size` instead.
`gM`	Whether to draw the major grid.
`gm`	Whether to draw the minor grid.
`gc`	Grid color, as a name or hex code.
`gl`	Grid line type (`lty`).
`boxes`	Whether to draw a border around the plot.
`bc`	Background color, as a name or hex code.
`pc`	Panel background color, as a name or hex code.
`lp`	Legend position.
`axis`	Axis label angle, as defined by `par("las")`.

Value

A ggplot2 theme object (class `theme`).

Examples

```
require("ggplot2")
if (require("pander")) {
  p <- ggplot(mtcars, aes(x = mpg, y = wt)) +
    geom_point()
  p + theme_pander()

  old_grid_color <- panderOptions("graph.grid.color")
  panderOptions("graph.grid.color", "red")
  p + theme_pander()
  panderOptions("graph.grid.color", old_grid_color)

  p <- ggplot(mtcars, aes(wt, mpg, colour = factor(cyl))) +
    geom_point()
  p + theme_pander() + scale_color_pander()

  ggplot(mpg, aes(x = class, fill = drv)) +
    geom_bar() +
    scale_fill_pander() +
    theme_pander()
}
```

---

theme_par *Theme based on base graphics parameters*

Description

Theme which uses the current "base" graphics parameter values from `graphics::par()`. Not all `par()` parameters are supported, and not all are relevant to ggplot2 themes.

Usage

```
theme_par(base_size = par()$ps, base_family = par()$family)
```

Arguments

`base_size`	base font size, given in pts.
`base_family`	base font family

Details

Currently this theme uses the values of the parameters: `"ps"`, `"family"`, `"fg"`, `"bg"`, `"col"`, `"adj"`, `"font"`, `"cex.axis"`, `"cex.lab"`, `"cex.main"`, `"cex.sub"`, `"col.axis"`, `"col.lab"`, `"col.main"`, `"col.sub"`, `"font.axis"`, `"font.lab"`, `"font.main"`, `"font.sub"`, `"las"`, `"lend"`, `"lheight"`, `"lty"`, `"mar"`, `"tcl"`, `"tck"`, `"xaxt"`, `"yaxt"`.

This theme does not translate the base graphics perfectly, so the graphs produced by it will not be identical to those produced by base graphics, most notably in the spacing of the margins.

Value

A ggplot2 theme object (class theme).

See Also

Other themes: theme_base(), theme_clean(), theme_foundation(), theme_igray(), theme_map(), theme_solid()

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am)

p + theme_par()

# theme changes with respect to values of par
old_par <- par(font = 2, col.lab = "red", fg = "white", bg = "black")
p + theme_par()
par(old_par)
```

---

theme_solarized *ggplot color themes based on the Solarized palette*

Description

See https://ethanschoonover.com/solarized/ for a description of the Solarized palette.

Usage

```
theme_solarized(base_size = 12, base_family = "", light = TRUE)

theme_solarized_2(base_size = 12, base_family = "", light = TRUE)
```

Arguments

base_size	base font size, given in pts.
base_family	base font family
light	If TRUE (the default), use the light theme; otherwise, use the dark theme.

Details

Plots made with this theme integrate seamlessly with the Solarized Beamer color theme. https://github.com/jrnold/beamercolorthemesolarized. There are two variations: theme_solarized() is similar to ggplot2::theme_bw(), while theme_solarized_2() is similar to ggplot2::theme_gray().

Value

A ggplot2 theme object (class theme).

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear)))

# Light version with different main accent colors
for (accent in names(ggthemes::ggthemes_data[["solarized"]][["accents"]])) {
  print(p + theme_solarized() + scale_colour_solarized(accent))
}

# Dark version
p + theme_solarized(light = FALSE) + scale_colour_solarized("blue")

# Alternative theme
p + theme_solarized_2(light = FALSE) + scale_colour_solarized("blue")
```

---

theme_solid — *Theme with nothing other than a background color*

Description

Theme that removes all non-geom elements (lines, text, etc.). Use it when only the geometric objects are desired. The base_family argument is ignored; it is kept for consistency with ggplot2::theme_grey().

Usage

```
theme_solid(base_size = 12, base_family = "", fill = NA)
```

Arguments

base_size	base font size, given in pts.
base_family	base font family
fill	The background color of the plot.

Value

A ggplot2 theme object (class theme).

See Also

Other themes: theme_base(), theme_clean(), theme_foundation(), theme_igray(), theme_map(), theme_par()

Examples

```
library("ggplot2")

ggplot(mtcars, aes(wt, mpg)) +
  geom_point() +
  theme_solid(fill = "white")

ggplot(mtcars, aes(wt, mpg)) +
  geom_point(color = "white") +
  theme_solid(fill = "black")
```

---

theme_stata *Themes based on Stata graph schemes*

Description

Themes based on Stata graph schemes

Usage

```
theme_stata(base_size = 11, base_family = "sans", scheme = NULL)
```

Arguments

base_size	base font size, given in pts.
base_family	base font family
scheme	A string, the Stata scheme. One of: "stcolor", "stcolor_alt", "stmono1", "stmono2", "stsj", "s2color", "s2mono", "s1color", "s1rcolor", "s1mono", "s2manual", "s1manual", or "sj". If NULL, the default, "s2color" is used and a deprecation message is issued; this default becomes "stcolor" in ggthemes 8.0.0.

Details

These themes approximate Stata schemes using the features ggplot2. The graphical models of Stata and ggplot2 differ in various ways that make an exact replication impossible (or more difficult than it is worth). Some features in Stata schemes not in ggplot2: defaults for specific graph types, different levels of titles, captions and notes. These themes also adopt some of the ggplot2 defaults, and more effort was made to match the colors and sizes of major elements than in matching the margins.

The schemes fall into two generations. "stcolor", "stcolor_alt", "stmono1", "stmono2" and "stsj" are the st family introduced in Stata 18, of which "stcolor" is Stata's current factory default: a white background, a dashed grid on both axes, horizontal y-axis labels, and a borderless legend beside the plot. The remaining schemes are the s1/s2 families that were the default through Stata 17.

Stata expresses text sizes as a percentage of graph height, while ggplot2 uses absolute points, so the two agree only at a particular graph size. The relative sizes here match Stata exactly;

base_size = 12.4 reproduces Stata's absolute sizes at its default 7.5 by 4.5 inch graph. Two further differences are not expressible in a ggplot2 theme: the number of legend columns (set by ggplot2::guide_legend() rather than the theme) and Stata's small default marker size (a geom default).

Value

A ggplot2 theme object (class theme).

References

https://www.stata.com/help.cgi?schemes

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am) +
  labs(
    title = "Graphs by car type",
    x = "Weight (lbs.)",
    y = "MPG"
  )

# The st family, Stata's default since Stata 18
# stcolor
p + theme_stata(scheme = "stcolor") + scale_colour_stata("stcolor")
# stcolor_alt, which puts the legend below the plot
p + theme_stata(scheme = "stcolor_alt") + scale_colour_stata("stcolor")
# stmono1
p + theme_stata(scheme = "stmono1") + scale_colour_stata("mono")
# stmono2
p + theme_stata(scheme = "stmono2") + scale_colour_stata("mono")
# stsj, the Stata Journal scheme
p + theme_stata(scheme = "stsj") + scale_colour_stata("mono")

# The s1/s2 families, Stata's defaults through Stata 17
# s2color
p + theme_stata(scheme = "s2color") + scale_colour_stata("s2color")
# s2mono
p + theme_stata(scheme = "s2mono") + scale_colour_stata("mono")
# s1color
p + theme_stata(scheme = "s1color") + scale_colour_stata("s1color")
# s1rcolor
p + theme_stata(scheme = "s1rcolor") + scale_colour_stata("s1rcolor")
# s1mono
p + theme_stata(scheme = "s1mono") + scale_colour_stata("mono")
```

theme_tufte *Tufte maximal data, minimal ink theme*

Description

Theme based on Chapter 6 'Data-Ink Maximization and Graphical Design' of Edward Tufte *The Visual Display of Quantitative Information*. No border, no axis lines, no grids. This theme works best in combination with `ggplot2::geom_rug()` or `geom_rangeframe()`.

Usage

```
theme_tufte(base_size = 11, base_family = "serif", ticks = TRUE)
```

Arguments

`base_size`	base font size, given in pts.
`base_family`	base font family
`ticks`	If `TRUE` (the default), show axis ticks.

Value

A ggplot2 theme object (class `theme`).

Note

The default font family is set to 'serif' as he uses serif fonts for labels in 'The Visual Display of Quantitative Information'. The serif font used by Tufte in his books is a variant of Bembo, while the sans serif font is Gill Sans. If these fonts are installed on your system, then you can use them with the package extrafont.

References

Tufte, Edward R. (2001) The Visual Display of Quantitative Information, Chapter 6.

See Also

Other tufte: `extended_range_breaks_()`, `geom_rangeframe()`, `geom_tufteboxplot()`

Examples

```
library("ggplot2")

p <- ggplot(mtcars, aes(x = wt, y = mpg)) +
  geom_point() +
  scale_x_continuous(breaks = extended_range_breaks()(mtcars$wt)) +
  scale_y_continuous(breaks = extended_range_breaks()(mtcars$mpg)) +
  ggtitle("Cars")
```

```
p + geom_rangeframe() + coord_cartesian(clip = "off") + theme_tufte()

p + geom_rug() + theme_tufte(ticks = FALSE)
```

---

theme_wsj *Wall Street Journal theme*

Description

Theme based on the plots in *The Wall Street Journal*.

Usage

```
theme_wsj(
  base_size = 12,
  color = "brown",
  base_family = "sans",
  title_family = "mono"
)
```

Arguments

base_size	base font size, given in pts.
color	A string, the background color of the plot. One of "gray", "green", "blue", "brown".
base_family	base font family
title_family	A string, the font family of the plot title.

Details

This theme should be used with scale_color_wsj().

Value

A ggplot2 theme object (class theme).

References

https://x.com/WSJGraphics

https://pinterest.com/wsjgraphics/wsj-graphics/

Examples

```
library("ggplot2")

p <- ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am) +
  ggtitle("Diamond Prices")
p + scale_colour_wsj("colors6", "") + theme_wsj()
# Use a gray background instead
p + scale_colour_wsj("colors6", "") + theme_wsj(color = "gray")
```

---

tremmel_shape_pal *Shape palette from Tremmel (1995) (discrete)*

Description

Based on experiments Tremmel (1995) suggests the following shape palettes:

Usage

```
tremmel_shape_pal(overlap = FALSE, alt = FALSE)
```

Arguments

overlap	Use an empty circle instead of a solid circle when n == 2.
alt	If TRUE, then when n == 3, use a solid circle, plus sign and empty triangle. Otherwise use a solid circle, empty circle, and empty triangle. Defaults to FALSE, the triple Tremmel's Experiment 1 actually measured; the TRUE triple is argued on feature-dimension grounds that Tremmel flags as not directly supported by the experiments.

Details

If two symbols, then use a solid circle and plus sign.

If three symbols, then use a solid circle, empty circle, and an empty triangle. However, that set of symbols does not satisfy the requirement that each symbol should differ from the other symbols in the same feature dimension. A set of three symbols that satisfies this is a circle (curvature), plus sign (number of terminators), triangle (line orientation).

This palette supports up to three values. If more than three groups of data, then separate the groups into different plots.

Value

A palette function. It takes the number of shapes n and returns an integer vector of n shape (pch) codes, and can be used as the palette argument of ggplot2::discrete_scale().

References

Tremmel, Lothar, (1995) "The Visual Separability of Plotting Symbols in Scatterplots" Journal of Computational and Graphical Statistics, https://www.jstor.org/stable/1390760

See Also

Other shapes: circlefill_shape_pal(), cleveland_shape_pal(), scale_shape_circlefill(), scale_shape_cleveland(), scale_shape_tremmel()

Examples

```
tremmel_shape_pal()(3)

# Alternative triple: solid circle, plus sign and empty triangle
tremmel_shape_pal(alt = TRUE)(3)

show_shapes(tremmel_shape_pal()(3))
```

---

wsj_pal *Wall Street Journal color palette (discrete)*

Description

The Wall Street Journal uses many different color palettes in its plots. This collects a few of them, but is by no means exhaustive. Collections of these plots can be found on the WSJ Graphics X (formerly Twitter) feed and Pinterest.

Usage

```
wsj_pal(palette = "colors6")
```

Arguments

palette	A string, the color palette to use. One of "rgby", "red_green", "black_green", "dem_rep", "colors6".

Value

A palette function. It takes the number of colors n and returns a character vector of n hex colors, and can be used as the palette argument of ggplot2::discrete_scale().

Palettes

The following palettes are defined:

- "rgby": red/green/blue/yellow theme.
- "red_green": green/red two-color scale for good/bad.

- "black_green": black-green 4-color scale for "very negative", "somewhat negative", "somewhat positive", "very positive".
- "dem_rep": Democrat/Republican/Undecided blue/red/gray scale.
- "colors6": red, blue, gold, green, orange, and black palette.

See Also

Other color wsj: scale_colour_wsj()

Examples

```
wsj_pal()(6)

scales::show_col(wsj_pal("rgby")(4))
```

Index

∗ **color calc**
calc_pal, 11
scale_fill_calc, 62
∗ **color canva**
canva_pal, 13
scale_colour_canva, 40
∗ **color colorblind**
colorblind_pal, 18
∗ **color economist**
economist_pal, 20
economist_seq_pal, 21
scale_colour_economist, 41
scale_colour_economist_c, 43
∗ **color excel**
excel_new_pal, 22
excel_pal, 23
scale_colour_excel_new, 44
scale_fill_excel, 64
∗ **color few**
few_pal, 25
scale_colour_few, 46
∗ **color fivethirtyeight**
fivethirtyeight_pal, 27
scale_colour_fivethirtyeight, 48
∗ **color gdocs**
gdocs_pal, 27
scale_fill_gdocs, 66
∗ **color hc**
hc_pal, 35
scale_colour_hc, 54
∗ **color numbers**
numbers_pal, 36
scale_fill_numbers, 67
∗ **color pander**
palette_pander, 37
scale_color_pander, 39
∗ **color ptol**
ptol_pal, 38
scale_colour_ptol, 55
∗ **color solarized**
scale_fill_solarized, 69
solarized_pal, 87
∗ **color stata**
scale_colour_stata, 57
stata_pal, 91
∗ **color tableau**
scale_colour_gradient2_tableau, 52
scale_colour_gradient_tableau, 49
scale_colour_tableau, 59
tableau_color_pal, 93
tableau_gradient_pal, 95
∗ **color wsj**
scale_colour_wsj, 61
wsj_pal, 122
∗ **datasets**
canva_palettes, 14
ggthemes_data, 34
∗ **geom tufte**
geom_rangeframe, 28
geom_tufteboxplot, 31
∗ **linetype stata**
scale_linetype_stata, 71
stata_linetype_pal, 90
∗ **shapes calc**
calc_shape_pal, 12
scale_shape_calc, 73
∗ **shapes few**
few_shape_pal, 26
scale_shape_few, 78
∗ **shapes stata**
scale_shape_stata, 79
stata_shape_pal, 92
∗ **shapes tableau**
scale_shape_tableau, 81
tableau_shape_pal, 96
∗ **shapes**
circlefill_shape_pal, 15
cleveland_shape_pal, 16

scale_shape_circlefill, 74
scale_shape_cleveland, 76
scale_shape_tremmel, 82
tremmel_shape_pal, 121
∗ **themes calc**
theme_calc, 98
∗ **themes economist**
theme_economist, 100
∗ **themes excel**
theme_excel, 102
theme_excel_new, 103
∗ **themes few**
theme_few, 104
∗ **themes fivethirtyeight**
theme_fivethirtyeight, 105
∗ **themes gdocs**
theme_gdocs, 107
∗ **themes hc**
theme_hc, 107
∗ **themes numbers**
theme_numbers, 112
∗ **themes pander**
theme_pander, 112
∗ **themes solarized**
theme_solarized, 115
∗ **themes stata**
theme_stata, 117
∗ **themes tufte**
theme_tufte, 119
∗ **themes wsj**
theme_wsj, 120
∗ **themes**
theme_base, 97
theme_clean, 99
theme_foundation, 105
theme_igray, 110
theme_map, 111
theme_par, 114
theme_solid, 116
∗ **tufte**
extended_range_breaks_, 23
geom_rangeframe, 28
geom_tufteboxplot, 31
theme_tufte, 119

aes(), *28*, *32*, *88*
annotation_borders(), *30*, *33*, *89*

bank_plot, 4
bank_plot(), *5*, *6*, *8*
bank_plot_multiscale, 5
bank_plot_multiscale(), *8*, *11*
bank_slopes, 6
bank_slopes(), *4*, *5*, *7*, *9–11*
bank_slopes_multiscale, 9
bank_slopes_multiscale(), *5*, *6*, *8*
base::format(), *86*

calc_pal, 11
calc_pal(), *64*
calc_shape_pal, 12
calc_shape_pal(), *73*, *74*
canva_pal, 13
canva_pal(), *41*
canva_palettes, *13*, 14, *40*, *41*
circlefill_shape_pal, 15
circlefill_shape_pal(), *17*, *76*, *78*, *84*, *122*
cleveland_shape_pal, 16
cleveland_shape_pal(), *16*, *76*, *78*, *84*, *122*
colorblind_pal, 18
colourblind_pal (colorblind_pal), 18

discrete_scale, *18*, *39*, *42*, *45*, *46*, *48*, *54*, *56*, *58*, *61*, *63*, *64*, *66*, *68*, *70*, *71*, *73*, *75*, *76*, *79*, *81*, *83*

economist_gradient_pal (economist_seq_pal), 21
economist_pal, 20
economist_pal(), *21*, *43*, *44*
economist_seq_pal, 21
economist_seq_pal(), *20*, *43*, *44*
excel_new_pal, 22
excel_new_pal(), *23*, *46*, *65*
excel_pal, 23
excel_pal(), *22*, *46*, *65*
extended_range_breaks (extended_range_breaks_), 23
extended_range_breaks_, 23
extended_range_breaks_(), *30*, *34*, *119*

few_pal, 25
few_pal(), *46*, *47*
few_shape_pal, 26
few_shape_pal(), *78*
fivethirtyeight_pal, 27
fivethirtyeight_pal(), *49*

fortify(), *28*, *32*, *88*

gdocs_pal, 27
gdocs_pal(), *67*
geom_rangeframe, 28
geom_rangeframe(), *24*, *34*, *119*
geom_tufteboxplot, 31
geom_tufteboxplot(), *24*, *30*, *119*
GeomRangeFrame *(*geom_rangeframe*)*, 28
GeomTufteboxplot *(*geom_tufteboxplot*)*, 31
ggplot(), *28*, *32*, *88*
ggplot2::continuous_scale(), *21*, *95*
ggplot2::coord_fixed(), *4*, *5*, *7*, *10*
ggplot2::discrete_scale(), *11*, *12*, *16*, *20–23*, *25–27*, *36–38*, *41*, *60*, *78*, *87*, *90–92*, *94*, *97*, *121*, *122*
ggplot2::geom_boxplot(), *34*
ggplot2::geom_map(), *111*
ggplot2::geom_rug(), *119*
ggplot2::ggplot_build(), *4*
ggplot2::guide_legend(), *118*
ggplot2::sec_axis(), *30*
ggplot2::stat_boxplot(), *32*, *90*
ggplot2::theme(), *101–103*
ggplot2::theme_bw(), *105*, *110*, *115*
ggplot2::theme_gray(), *105*, *106*, *110*, *115*
ggplot2::theme_grey(), *116*
ggthemes_data, 34
graphics::par(), *84*, *85*, *114*
guides(), *19*, *40*, *42*, *43*, *45*, *47*, *49*, *55*, *57*, *58*, *62*, *63*, *65*, *67*, *69*, *70*

hc_pal, 35
hc_pal(), *54*, *55*, *108*

key glyphs, *29*, *33*, *89*

lambda, *18*, *19*, *39*, *40*, *42*, *45–49*, *54–58*, *61–77*, *79–83*
lattice::banking(), *8*
layer geom, *88*
layer position, *29*, *32*, *89*
layer stat, *29*
layer(), *29*, *33*, *89*

numbers_pal, 36
numbers_pal(), *69*

palette_pander, 37
palette_pander(), *40*

ptol_pal, 38
ptol_pal(), *55*, *57*

scale_color_calc *(*scale_fill_calc*)*, 62
scale_color_canva *(*scale_colour_canva*)*, 40
scale_color_colorblind *(*colorblind_pal*)*, 18
scale_color_continuous_tableau *(*scale_colour_gradient_tableau*)*, 49
scale_color_economist *(*scale_colour_economist*)*, 41
scale_color_economist(), *101*
scale_color_economist_c *(*scale_colour_economist_c*)*, 43
scale_color_economist_ordinal *(*scale_colour_economist_c*)*, 43
scale_color_excel *(*scale_fill_excel*)*, 64
scale_color_excel_new *(*scale_colour_excel_new*)*, 44
scale_color_few *(*scale_colour_few*)*, 46
scale_color_fivethirtyeight *(*scale_colour_fivethirtyeight*)*, 48
scale_color_gdocs *(*scale_fill_gdocs*)*, 66
scale_color_gradient2_tableau *(*scale_colour_gradient2_tableau*)*, 52
scale_color_gradient_tableau *(*scale_colour_gradient_tableau*)*, 49
scale_color_hc *(*scale_colour_hc*)*, 54
scale_color_numbers *(*scale_fill_numbers*)*, 67
scale_color_pander, 39
scale_color_pander(), *37*
scale_color_ptol *(*scale_colour_ptol*)*, 55
scale_color_solarized *(*scale_fill_solarized*)*, 69
scale_color_stata *(*scale_colour_stata*)*, 57
scale_color_tableau *(*scale_colour_tableau*)*, 59
scale_color_tableau(), *19*
scale_color_wsj *(*scale_colour_wsj*)*, 61
scale_color_wsj(), *120*
scale_colour_calc *(*scale_fill_calc*)*, 62
scale_colour_canva, 40

scale_colour_canva(), *13*
scale_colour_colorblind (colorblind_pal), 18
scale_colour_colourblind (colorblind_pal), 18
scale_colour_continuous_tableau (scale_colour_gradient_tableau), 49
scale_colour_economist, 41
scale_colour_economist(), *20, 21*, *43, 44*
scale_colour_economist_c, 43
scale_colour_economist_c(), *20, 21*, *43*
scale_colour_economist_ordinal (scale_colour_economist_c), 43
scale_colour_excel (scale_fill_excel), 64
scale_colour_excel(), *102*
scale_colour_excel_new, 44
scale_colour_excel_new(), *22, 23*, *65*, *103*
scale_colour_few, 46
scale_colour_few(), *26*
scale_colour_fivethirtyeight, 48
scale_colour_fivethirtyeight(), *27*
scale_colour_gdocs (scale_fill_gdocs), 66
scale_colour_gradient2_tableau, 52
scale_colour_gradient2_tableau(), *49*, *51*, *59, 60*, *94*, *96*
scale_colour_gradient_tableau, 49
scale_colour_gradient_tableau(), *52, 53*, *59, 60*, *94*, *96*
scale_colour_hc, 54
scale_colour_hc(), *36*
scale_colour_numbers (scale_fill_numbers), 67
scale_colour_pander (scale_color_pander), 39
scale_colour_ptol, 55
scale_colour_ptol(), *38*
scale_colour_solarized (scale_fill_solarized), 69
scale_colour_stata, 57
scale_colour_stata(), *91*
scale_colour_tableau, 59
scale_colour_tableau(), *49*, *51–53*, *94*, *96*
scale_colour_wsj, 61
scale_colour_wsj(), *123*
scale_fill_calc, 62
scale_fill_calc(), *12*
scale_fill_canva (scale_colour_canva), 40
scale_fill_colorblind (colorblind_pal), 18
scale_fill_colourblind (colorblind_pal), 18
scale_fill_continuous_tableau (scale_colour_gradient_tableau), 49
scale_fill_economist (scale_colour_economist), 41
scale_fill_economist_c (scale_colour_economist_c), 43
scale_fill_economist_ordinal (scale_colour_economist_c), 43
scale_fill_excel, 64
scale_fill_excel(), *22, 23*, *46*
scale_fill_excel_new (scale_colour_excel_new), 44
scale_fill_few (scale_colour_few), 46
scale_fill_fivethirtyeight (scale_colour_fivethirtyeight), 48
scale_fill_gdocs, 66
scale_fill_gdocs(), *27*
scale_fill_gradient2_tableau (scale_colour_gradient2_tableau), 52
scale_fill_gradient_tableau (scale_colour_gradient_tableau), 49
scale_fill_hc (scale_colour_hc), 54
scale_fill_numbers, 67
scale_fill_numbers(), *37*
scale_fill_pander (scale_color_pander), 39
scale_fill_ptol (scale_colour_ptol), 55
scale_fill_solarized, 69
scale_fill_solarized(), *87*
scale_fill_stata (scale_colour_stata), 57
scale_fill_tableau (scale_colour_tableau), 59
scale_fill_wsj (scale_colour_wsj), 61
scale_linetype_stata, 71
scale_linetype_stata(), *90*
scale_shape_calc, 73

scale_shape_calc(), *12*
scale_shape_circlefill, 74
scale_shape_circlefill(), *16, 17*, *78*, *84*, *122*
scale_shape_cleveland, 76
scale_shape_cleveland(), *16, 17*, *76*, *84*, *122*
scale_shape_few, 78
scale_shape_few(), *26*
scale_shape_stata, 79
scale_shape_stata(), *92*
scale_shape_tableau, 81
scale_shape_tableau(), *74*, *97*
scale_shape_tremmel, 82
scale_shape_tremmel(), *16, 17*, *76*, *78*, *122*
scales::dichromat_pal(), *19*
scales::show_col(), *85*
show_linetypes, 84
show_linetypes(), *85*
show_shapes, 85
show_shapes(), *85*
smart_digits, 86
smart_digits_format *(*smart_digits*)*, 86
solarized_pal, 87
solarized_pal(), *69*, *71*
stat_fivenumber, 88
stat_fivenumber(), *32*
stata_linetype_pal, 90
stata_linetype_pal(), *71, 72*
stata_pal, 91
stata_pal(), *35*, *57*, *59*
stata_shape_pal, 92
stata_shape_pal(), *79, 80*
StatFivenumber *(*stat_fivenumber*)*, 88
stats::uniroot(), *8*

tableau_color_pal, 93
tableau_color_pal(), *51*, *53*, *60*, *95, 96*
tableau_colour_pal *(*tableau_color_pal*)*, 93
tableau_div_gradient_pal *(*tableau_gradient_pal*)*, 95
tableau_gradient_pal, 95
tableau_gradient_pal(), *51*, *53*, *60*, *94*
tableau_seq_gradient_pal *(*tableau_gradient_pal*)*, 95
tableau_shape_pal, 96
tableau_shape_pal(), *81, 82*
theme_base, 97
theme_base(), *99*, *106*, *110, 111*, *115, 116*
theme_calc, 98
theme_calc(), *64*, *74*
theme_clean, 99
theme_clean(), *98*, *106*, *110, 111*, *115, 116*
theme_economist, 100
theme_economist(), *21*, *43*, *100*, *106*
theme_economist_white *(*theme_economist*)*, 100
theme_economist_white(), *100*
theme_excel, 102
theme_excel(), *103*, *106*
theme_excel_new, 103
theme_excel_new(), *102*
theme_few, 104
theme_fivethirtyeight, 105
theme_fivethirtyeight(), *49*
theme_foundation, 105
theme_foundation(), *98, 99*, *110, 111*, *115, 116*
theme_gdocs, 107
theme_gdocs(), *67*
theme_hc, 107
theme_hc(), *54*, *106*
theme_igray, 110
theme_igray(), *98, 99*, *106*, *111*, *115, 116*
theme_map, 111
theme_map(), *98, 99*, *106*, *110*, *115, 116*
theme_numbers, 112
theme_numbers(), *69*
theme_pander, 112
theme_pander(), *40*
theme_par, 114
theme_par(), *98, 99*, *106*, *110, 111*, *116*
theme_solarized, 115
theme_solarized(), *115*
theme_solarized_2 *(*theme_solarized*)*, 115
theme_solid, 116
theme_solid(), *98, 99*, *106*, *110, 111*, *115*
theme_stata, 117
theme_tufte, 119
theme_tufte(), *24*, *30*, *34*
theme_wsj, 120
theme_wsj(), *61*
tibble::tibble(), *10*
tremmel_shape_pal, 121
tremmel_shape_pal(), *16, 17*, *76*, *78*, *84*

wsj_pal, 122

wsj_pal(), *61*, *62*