

# Package ‘ggthemes’

November 30, 2025

**Version** 5.2.0

**Title** Extra Themes, Scales and Geoms for 'ggplot2'

**Depends** R (>= 3.3.0)

**Imports** ggplot2 (>= 3.0.0), graphics, grid, lifecycle, methods, purrr,  
scales, stringr, tibble

**Suggests** dplyr, covr, extrafont, glue, knitr, lattice, lintr, maps,  
mapproj, pander, quantreg, rlang, rmarkdown, spelling, testthat  
(>= 3.2.0), tidyr, vdiff, 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>,  
<http://jrnold.github.io/ggthemes/>

**BugReports** <https://github.com/jrnold/ggthemes/issues>

**RoxygenNote** 7.3.3

**LazyData** true

**Language** en-US

**Encoding** UTF-8

**Config/testthat/edition** 3

**Config/Needs/website** pkgdown

**NeedsCompilation** no

**Author** Jeffrey B. Arnold [aut, cre] (<<https://orcid.org/0000-0001-9953-3904>>),  
Gergely Daroczi [ctb],  
Bo Werth [ctb],  
Brian Weitzner [ctb],  
Joshua Kunst [ctb],

Baptise Auguie [ctb],  
 Bob Rudis [ctb],  
 Hadley Wickham [ctb] (Code from the ggplot2 package.),  
 Justin Talbot [ctb] (Code from the labeling package),  
 Joshua London [ctb]

**Maintainer** Jeffrey B. Arnold <jeffrey.arnold@gmail.com>

**Repository** CRAN

**Date/Publication** 2025-11-30 17:00:12 UTC

## Contents

bank_slopes . . . . .	4
calc_pal . . . . .	6
calc_shape_pal . . . . .	6
canva_pal . . . . .	7
canva_palettes . . . . .	8
circlefill_shape_pal . . . . .	9
cleveland_shape_pal . . . . .	10
colorblind_pal . . . . .	11
economist_pal . . . . .	13
excel_new_pal . . . . .	14
excel_pal . . . . .	14
extended_range_breaks_ . . . . .	15
few_pal . . . . .	16
few_shape_pal . . . . .	17
fivethirtyeight_pal . . . . .	17
gdocs_pal . . . . .	18
geom_rangeframe . . . . .	18
geom_tufteboxplot . . . . .	21
ggthemes_data . . . . .	24
hc_pal . . . . .	25
palette_pander . . . . .	25
ptol_pal . . . . .	26
scale_color_pander . . . . .	27
scale_colour_canva . . . . .	28
scale_colour_economist . . . . .	29
scale_colour_excel_new . . . . .	30
scale_colour_few . . . . .	32
scale_colour_fivethirtyeight . . . . .	33
scale_colour_gradient2_tableau . . . . .	35
scale_colour_gradient_tableau . . . . .	37
scale_colour_hc . . . . .	39
scale_colour_ptol . . . . .	40
scale_colour_stata . . . . .	42
scale_colour_tableau . . . . .	43
scale_colour_wsj . . . . .	45
scale_fill_calc . . . . .	46

scale_fill_excel . . . . .	48
scale_fill_gdocs . . . . .	49
scale_fill_solarized . . . . .	51
scale_linetype_stata . . . . .	52
scale_shape_calc . . . . .	54
scale_shape_circlefill . . . . .	55
scale_shape_cleveland . . . . .	57
scale_shape_few . . . . .	58
scale_shape_stata . . . . .	59
scale_shape_tableau . . . . .	60
scale_shape_tremmel . . . . .	62
show_linetypes . . . . .	64
show_shapes . . . . .	65
smart_digits . . . . .	65
solarized_pal . . . . .	66
stata_linetype_pal . . . . .	67
stata_pal . . . . .	67
stata_shape_pal . . . . .	68
stat_fivenumber . . . . .	68
tableau_color_pal . . . . .	70
tableau_gradient_pal . . . . .	72
tableau_shape_pal . . . . .	73
theme_base . . . . .	74
theme_calc . . . . .	75
theme_clean . . . . .	75
theme_economist . . . . .	76
theme_excel . . . . .	78
theme_excel_new . . . . .	79
theme_few . . . . .	80
theme_fivethirtyeight . . . . .	81
theme_foundation . . . . .	81
theme_gdocs . . . . .	82
theme_hc . . . . .	82
theme_igray . . . . .	85
theme_map . . . . .	86
theme_pander . . . . .	86
theme_par . . . . .	88
theme_solarized . . . . .	89
theme_solid . . . . .	90
theme_stata . . . . .	91
theme_tufte . . . . .	92
theme_wsj . . . . .	93
tremmel_shape_pal . . . . .	94
wsj_pal . . . . .	95

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 Herr and Argwala (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 = FALSE, weight = NULL, method = c("ms", "as"), ...)
```

**Arguments**

x	x values
y	y values
cull	logical. Remove all slopes of 0 or Inf.
weight	No longer used, but kept for backwards compatibility.
method	One of 'ms' (Median Absolute Slope) or 'as' (Average Absolute Slope). Other options are no longer supported, and will use 'ms' instead with a warning.
...	No longer used, but kept for backwards compatibility.

**Value**

numeric The aspect ratio (x, y).

**Methods**

As written, all of these methods calculate the aspect ratio (x / y), but bank\_slopes will return (y / x) to be compatible with `link[ggplot2]{coord_fixed()}`.

**Median Absolute Slopes Banking**

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

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

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

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

Heer and Agrawala (2006) and Cleveland discuss several other methods including average (weighted) orientation, and global and local orientation resolution. These are no longer implemented in this function. In general, either the median or average absolute slopes will produce reasonable results without requiring optimization.

## 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

[banking\(\)](#)

## 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)
## Using culling
## Average Absolute Slope
bank_slopes(x, y, method = "as")
```

---

calc_pal	<i>Calc color palette (discrete)</i>
----------	--------------------------------------

---

**Description**

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

**Usage**

```
calc_pal()
```

**See Also**

Other colour calc: [scale\\_fill\\_calc\(\)](#)

**Examples**

```
library("scales")  
  
show_col(calc_pal()(12))
```

---

calc_shape_pal	<i>Calc shape palette (discrete)</i>
----------------	--------------------------------------

---

**Description**

Shape palette based on the shapes used in LibreOffice Calc.

**Usage**

```
calc_shape_pal()
```

**See Also**

Other shapes calc: [scale\\_shape\\_calc\(\)](#)

**Examples**

```
## Not run:  
library("ggplot2")  
show_shapes(calc_shape_pal()(13))  
  
## End(Not run)
```

---

canva_pal	<i>Canva.com color palettes</i>
-----------	---------------------------------

---

**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.

**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	<i>150 Color Palettes from Canva</i>
----------------	--------------------------------------

---

## Description

150 four-color palettes by the [canva.com](https://canva.com) design school. These palettes were derived from photos and "impactful websites".

## Usage

```
canva_palettes
```

## Format

A named list of character vector. 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

‘r lifecycle::badge("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()
```

## 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\(\)](#)

---

cleveland\_shape\_pal     *Shape palette from Cleveland "Elements of Graphing Data" (discrete).*

---

### Description

Shape palettes for overlapping and non-overlapping points.

### Usage

```
cleveland_shape_pal(overlap = TRUE)
```

### Arguments

overlap                logical Use the scale for overlapping points?

### 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 relies on pattern discrimination, while the symbols for non-overlapping data vary the amount of fill. This palette attempts to create these palettes. However, I found that these were hard to replicate. Using the R shapes and unicode fonts: the symbols can vary in size, they are dependent of the fonts used, and there does not exist a unicode symbol for a circle with a vertical line. If someone can improve this palette, please let me know.

Following Tremmel (1995), I replace the circle with a vertical line with an encircled plus sign.

The palette `cleveland_shape_pal()` supports up to five values.

### 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

```
###     (discrete).

## Not run:
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)

## End(Not run)

```

---

colorblind\_pal

*Colorblind Color Palette (Discrete) and Scales*


---

## Description

An eight-color colorblind safe qualitative discrete palette.

‘r lifecycle::badge("deprecated")‘

## Usage

```

colorblind_pal()

colourblind_pal()

scale_colour_colourblind(...)

scale_colour_colorblind(...)

scale_color_colorblind(...)

scale_fill_colorblind(...)

scale_fill_colourblind(...)

```

## 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 `?plot-math` 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

## References

Chang, W. "[Cookbook for R](#)"  
<https://jfly.iam.u-tokyo.ac.jp/color>

## See Also

The **dichromat** package, [dichromat\\_pal\(\)](#), and [scale\\_color\\_tableau\(\)](#) for other colorblind palettes.

## Examples

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

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

---

economist_pal	<i>Economist color palette (discrete)</i>
---------------	---

---

## Description

The hues in the palette are blues, grays, and greens. Red is not included in these palettes and should be used to indicate important data.

## Usage

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

## Arguments

fill	Use the fill palette.
------	-----------------------

## See Also

Other colour economist: [scale\\_colour\\_economist\(\)](#)

## Examples

```
library("scales")

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

---

excel_new_pal	<i>Excel (current versions) color palettes (discrete)</i>
---------------	---

---

**Description**

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

**Usage**

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

**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 Theme", "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 2007-2010", "Orange Red", "Orange", "Paper", "Red Orange", "Red Violet", "Red", "Slipstream", "Violet II", "Violet", "Yellow Orange", "Yellow"
-------	---

**See Also**

Other colour excel: [excel\\_pal\(\)](#), [scale\\_colour\\_excel\\_new\(\)](#), [scale\\_fill\\_excel\(\)](#)

**Examples**

```
library("scales")

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

---

excel_pal	<i>Excel 97 ugly color palettes (discrete)</i>
-----------	--

---

**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.

**See Also**

Other colour 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**

<code>dmin</code>	minimum of the data range
<code>dmax</code>	maximum of the data range
<code>n</code>	desired number of breaks
<code>Q</code>	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. `scales_extended_range_breaks` uses the conventions of the **scales** package, and returns a function.

**Value**

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

A function which returns breaks given a vector.

**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.

---

<code>few_pal</code>	<i>Color Palettes Few "Show Me the Numbers"</i>
----------------------	---

---

**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

**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.



## 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 colour 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	<i>Shape palette from "Show Me the Numbers" (discrete)</i>
---------------	--

---

## 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()
```

## References

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

---

fivethirtyeight_pal	<i>FiveThirtyEight color palette</i>
---------------------	--------------------------------------

---

## Description

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

## Usage

```
fivethirtyeight_pal()
```

**See Also**

Other colour 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()
```

**See Also**

Other colour 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 = "b1",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

**Arguments**

mapping	Set of aesthetic mappings created by <a href="#">aes()</a> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	<p>The statistical transformation to use on the data for this layer. When using a <code>geom_*()</code> function to construct a layer, the <code>stat</code> argument can be used to override the default coupling between geoms and stats. The <code>stat</code> argument accepts the following:</p> <ul style="list-style-type: none"> <li>• A Stat ggproto subclass, for example <code>StatCount</code>.</li> <li>• A string naming the stat. To give the stat as a string, strip the function name of the <code>stat_</code> prefix. For example, to use <code>stat_count()</code>, give the stat as <code>"count"</code>.</li> <li>• For more information and other ways to specify the stat, see the <a href="#">layer stat</a> documentation.</li> </ul>
position	<p>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 <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> <li>• The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position.</li> <li>• A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as <code>"jitter"</code>.</li> </ul>

	<ul style="list-style-type: none"> <li>For more information and other ways to specify the position, see the <a href="#">layer position</a> documentation.</li> </ul>
...	<p>Other arguments passed on to <a href="#">layer()</a>'s <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored.</p> <ul style="list-style-type: none"> <li>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, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an <b>Aesthetics</b> section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. 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.</li> <li>When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept.</li> <li>Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept.</li> <li>The <code>key_glyph</code> argument of <a href="#">layer()</a> may also be passed on through ... This can be one of the functions described as <a href="#">key glyphs</a>, to change the display of the layer in the legend.</li> </ul>
<code>sides</code>	A string that controls which sides of the plot the frames appear on. It can be set to a string containing any of 'trbl', for top, right, bottom, and left.
<code>na.rm</code>	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.
<code>show.legend</code>	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.
<code>inherit.aes</code>	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. <a href="#">annotation_borders()</a> .

## Details

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

## 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()`

## 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
```

---

`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 functions 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 <a href="#">aes()</a> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
stat	The statistical transformation to use on the data for this layer, as a string. The default ( <code>stat = 'fivenumber'</code> ) calls <a href="#">stat_fivenumber</a> and produces whiskers that extend from the interquartile range to the extremes of the data; specifying <a href="#">stat_boxplot</a> 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	<p>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:</p> <ul style="list-style-type: none"> <li>• The result of calling a position function, such as <a href="#">position_jitter()</a>. This method allows for passing extra arguments to the position.</li> <li>• A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <a href="#">position_jitter()</a>, give the position as <code>"jitter"</code>.</li> <li>• For more information and other ways to specify the position, see the <a href="#">layer position</a> documentation.</li> </ul>
outlier.colour	colour for outlying points
outlier.shape	shape of outlying points
outlier.size	size of outlying points
outlier.stroke	stroke for outlying points

voffset	controls the size of the gap in the line representing the median when <code>median.type = 'line'</code> . This is a fraction of the range of <code>y</code> .
hoffset	controls how much the interquartile line is offset from the whiskers when <code>median.type = 'line'</code> . This is a fraction of the range of <code>x</code> .
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. <a href="#">annotation_borders()</a> .
median.type	If 'point', then the median is represented by a point, and the interquartile range by a gap in the line. If <code>median.type='line'</code> , then the interquartile range is represented by a line, possibly offset, and the median by a gap in the line.
whisker.type	If 'line', then whiskers are represented by lines. If 'point', then whiskers are represented by points at <code>ymin</code> and <code>ymax</code> .
...	Other arguments passed on to <a href="#">layer()</a> 's <code>params</code> argument. These arguments broadly fall into one of 4 categories below. Notably, further arguments to the <code>position</code> argument, or aesthetics that are required can <i>not</i> be passed through ... Unknown arguments that are not part of the 4 categories below are ignored. <ul style="list-style-type: none"> <li>• 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, <code>colour = "red"</code> or <code>linewidth = 3</code>. The geom's documentation has an <b>Aesthetics</b> section that lists the available options. The 'required' aesthetics cannot be passed on to the <code>params</code>. 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.</li> <li>• When constructing a layer using a <code>stat_*()</code> function, the ... argument can be used to pass on parameters to the geom part of the layer. An example of this is <code>stat_density(geom = "area", outline.type = "both")</code>. The geom's documentation lists which parameters it can accept.</li> <li>• Inversely, when constructing a layer using a <code>geom_*()</code> function, the ... argument can be used to pass on parameters to the stat part of the layer. An example of this is <code>geom_area(stat = "density", adjust = 0.5)</code>. The stat's documentation lists which parameters it can accept.</li> <li>• The <code>key_glyph</code> argument of <a href="#">layer()</a> may also be passed on through ... This can be one of the functions described as <a href="#">key glyphs</a>, to change the display of the layer in the legend.</li> </ul>

## Aesthetics

- `x` [required]
- `y` [required]

- colour
- size
- 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

[geom\\_boxplot\(\)](#)

Other geom tufte: [geom\\_rangeframe\(\)](#)

## 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
p + geom_tufteboxplot(median.type = "line", hoffset = 0, width = 3)
# 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

The ggthemes environment contains various values used in themes and palettes. This is undocumented and subject to change.

## Usage

ggthemes\_data



**Format**

A list object.

---

hc_pal	<i>Highcharts color palette (discrete)</i>
--------	--

---

**Description**

The Highcharts uses many different color palettes in its plots. This collects a few of them.

**Usage**

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

**Arguments**

palette	character The name of the Highcharts theme to use. One of "default", or "darkunica".
---------	--

**See Also**

Other colour hc: [scale\\_colour\\_hc\(\)](#)

---

palette_pander	<i>Color palette from the pander package</i>
----------------	--

---

**Description**

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

**Usage**

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

**Arguments**

n	number of colors. This palette supports up to eight colors.
random_order	if the palette should be reordered randomly before rendering each plot to get colorful images

**See Also**

Other colour pander: [scale\\_color\\_pander\(\)](#)

### Examples

```
## Not run:  
palette_pander(TRUE)  
  
## End(Not run)
```

---

ptol\_pal

*Color Palettes from Paul Tol's "Colour Schemes"*

---

### Description

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](<https://tradeblotter.wordpress.com/2013/02/28/the-paul-tol-21-color-salute/>)

### References

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

### See Also

Other colour 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	<i>Color scale from the pander package</i>
--------------------	--

---

## Description

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

## 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 `?plot-math` 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

### See Also

`theme_pander()`

Other colour pander: `palette_pander()`

---

scale_colour_canva	<i>Discrete color scale using canva.com color palettes</i>
--------------------	--

---

### 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 `discrete_scale()`.

`palette` Palette name. See the names of `canva_palettes()` for valid names.

---

`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 `?plot-math` 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

## See Also

`theme_economist()` for examples.

Other colour economist: `economist_pal()`

---

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 Theme", ...)
```

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

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

## 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 Theme", "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 2007-2010", "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 `?plot-math` 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

### See Also

Other colour 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	<i>Color scales from Few's "Practical Rules for Using Color in Charts"</i>
------------------	--

---

### Description

See [few\\_pal\(\)](#).

### Usage

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

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

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

### Arguments

palette	One of
...	Arguments passed on to <a href="#">discrete_scale</a>
breaks	One of: <ul style="list-style-type: none"> <li>• NULL for no breaks</li> <li>• <a href="#">waiver()</a> for the default breaks (the scale limits)</li> <li>• A character vector of breaks</li> <li>• A function that takes the limits as input and returns breaks as output. Also accepts rlang <a href="#">lambda</a> function notation.</li> </ul>
limits	One of: <ul style="list-style-type: none"> <li>• NULL to use the default scale values</li> <li>• A character vector that defines possible values of the scale and their order</li> </ul>



- 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 `?plot-math` 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

## See Also

Other colour few: [few\\_pal\(\)](#)

---

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 `?plot-math` 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

### See Also

[theme\\_fivethirtyeight\(\)](#) for examples.

Other colour fivethirtyeight: [fivethirtyeight\\_pal\(\)](#)

---

scale\_colour\_gradient2\_tableau

*Tableau diverging colour 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",
  ...,
  na.value = "grey50",
  guide = "colourbar"
)
```

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

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

**Arguments**

palette	Palette name. "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 <code>tableau_gradient_pal</code> .
na.value	Colour to use for missing values
guide	Type of legend. Use 'colourbar' for continuous colour bar, or 'legend' for discrete colour legend.

**See Also**

Other colour 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_gradient_tableau`*Tableau sequential colour 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_fill_continuous_tableau(  
  palette = "Blue",  
  ...,  
  na.value = "grey50",  
  guide = "colourbar"  
)
```

**Arguments**

palette	Palette name. "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 <code>tableau_gradient_pal</code> .
na.value	Colour to use for missing values
guide	Type of legend. Use 'colourbar' for continuous colour bar, or 'legend' for discrete colour legend.

**See Also**

Other colour 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_hc	<i>Highcharts color and fill scales</i>
-----------------	---

---

## Description

Colour 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** character The name of the Highcharts theme to use. One of "default", or "darkunica".

**...** 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 ?plot-math 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

### See Also

Other colour hc: [hc\\_pal\(\)](#)

---

scale\_colour\_ptol

*Color Scales from Paul Tol's "Colour Schemes"*

---

### Description

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 `?plot-math` 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

## See Also

Other colour 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	<i>Stata color scales</i>
--------------------	---------------------------

---

## Description

See [stata\\_pal\(\)](#) for details.

## Usage

```
scale_colour_stata(scheme = "s2color", ...)
```

```
scale_fill_stata(scheme = "s2color", ...)
```

```
scale_color_stata(scheme = "s2color", ...)
```

## Arguments

**scheme** character. One of "s2color", "s1rcolor", "s1color", or "mono".

**...** 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 `?plot-math` 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

---

`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 <a href="#">tableau_color_pal()</a> for available palettes.
type	Palette type. One of "regular", "sequential", or "diverging". See <a href="#">tableau_color_pal()</a> .
direction	If 1, the default, then use the original order of colors. If -1, then reverse the order.
...	Other arguments passed on to <a href="#">discrete_scale()</a> .

### See Also

[tableau\\_color\\_pal\(\)](#) for references.

Other colour 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	<i>Wall Street Journal color and fill scales</i>
------------------	--

---

## Description

Colour 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** character The color palette to use: . "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 `?plot-math` 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

### See Also

Other colour wsj: [wsj\\_pal\(\)](#)

---

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 `?plot-math` 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

## See Also

See [theme\\_calc\(\)](#) for examples.

Other colour calc: [calc\\_pal\(\)](#)

---

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 `?plot-math` 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

### See Also

Other colour 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 `?plot-math` 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

**See Also**

See [theme\\_gdocs\(\)](#) for examples.

Other colour gdocs: [gdocs\\_pal\(\)](#)

---

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            character 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 `?plot-math` 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

## See Also

Other solarized colour: [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 `?plot-math` 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

**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(...)
```

**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 `?plot-math` 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

**See Also**

[theme\\_calc\(\)](#) for examples.

Other shapes calc: [calc\\_shape\\_pal\(\)](#)

---

scale\_shape\_circlefill

*Filled Circle Shape palette (discrete)*

---

**Description**

‘r lifecycle::badge("deprecated")‘

**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 `?plot-math` 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

### 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\(\)](#)

---

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, ...)
```

### 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 `?plot-math` 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

## 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\(\)](#)

---

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 [discrete\\_scale\(\)](#) parameters.

**References**

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

**See Also**

[scale\\_shape\\_few\(\)](#) for the shape palette that this scale uses.

---

scale_shape_stata	<i>Stata shape scale</i>
-------------------	--------------------------

---

**Description**

See [stata\\_shape\\_pal\(\)](#) for details.

**Usage**

```
scale_shape_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 `?plot-math` 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

## Examples

```
## Not run:
library("ggplot2")

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

## End(Not run)
```

---

scale_shape_tableau	<i>Tableau shape scales</i>
---------------------	-----------------------------

---

## Description

See [tableau\\_shape\\_pal\(\)](#) for details.

**Usage**

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

**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 `?plot-math` 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

## See Also

Other shape tableau: [tableau\\_shape\\_pal\(\)](#)

## Examples

```
## Not run:
library("ggplot2")

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

## End(Not run)
```

---

scale_shape_tremmel	<i>Shape scales from Tremmel (1995)</i>
---------------------	---

---

## Description

Shape scales from Tremmel (1995)

## Usage

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

## Arguments

<code>overlap</code>	use an empty circle instead of a solid circle when <code>n == 2</code> .
<code>alt</code>	If TRUE, then when <code>n == 3</code> , use a solid circle, plus sign and empty triangle. Otherwise use a solid circle, empty circle, and empty triangle.
<code>...</code>	Arguments passed on to <a href="#">discrete_scale</a>

`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 `?plot-math` 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

## 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 <a href="#">par()</a> .
labels	Label each line with its linetype (lty) value.

### Value

This function called for the side effect of creating a plot. It returns `linetypes`.

### See Also

[show\\_col\(\)](#), [show\\_linetypes\(\)](#)

### Examples

```
library("scales")

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



---

show_shapes	<i>Show shapes</i>
-------------	--------------------

---

**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 <a href="#">par()</a> .
labels	Include the plotting character value of the symbol.

**Value**

This function called for the side effect of creating a plot. It returns shapes.

**See Also**

[show\\_col\(\)](#), [show\\_linetypes\(\)](#)

**Examples**

```
library("scales")

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

---

smart_digits	<i>Format numbers with automatic number of digits</i>
--------------	---

---

**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 `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, Baptise Auguie, Jeffrey B. Arnold

**References**

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

---

solarized_pal	<i>Solarized color palette (discrete)</i>
---------------	---

---

**Description**

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

**Usage**

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

**Arguments**

`accent`                character Starting color.

**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 solarized colour: `scale_fill_solarized()`

**Examples**

```
library("scales")

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

---

stata_linetype_pal	<i>Stata linetype palette (discrete)</i>
--------------------	--

---

**Description**

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

**Usage**

```
stata_linetype_pal()
```

**See Also**

[scale\\_linetype\\_stata\(\)](#)

Other linetype stata: [scale\\_linetype\\_stata\(\)](#)

---

stata_pal	<i>Stata color palettes (discrete)</i>
-----------	--

---

**Description**

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

**Usage**

```
stata_pal(scheme = "s2color")
```

**Arguments**

scheme                      character. One of "s2color", "s1rcolor", "s1color", or "mono".

**Details**

All these palettes support up to 15 values.

**Examples**

```
library("scales")

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	<i>Stata shape palette (discrete)</i>
-----------------	---------------------------------------

---

**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()
```

**See Also**

See [scale\\_shape\\_stata\(\)](#) for examples.

---

stat_fivenumber	<i>Calculate components of a five-number summary</i>
-----------------	--

---

**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 <a href="#">aes()</a> . If specified and <code>inherit.aes = TRUE</code> (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	<p>The data to be displayed in this layer. There are three options:</p> <p>If <code>NULL</code>, the default, the data is inherited from the plot data as specified in the call to <a href="#">ggplot()</a>.</p> <p>A <code>data.frame</code>, or other object, will override the plot data. All objects will be fortified to produce a data frame. See <a href="#">fortify()</a> for which variables will be created.</p> <p>A function will be called with a single argument, the plot data. The return value must be a <code>data.frame</code>, and will be used as the layer data. A function can be created from a formula (e.g. <code>~ head(.x, 10)</code>).</p>
geom	<p>The geometric object to use to display the data for this layer. When using a <code>stat_*()</code> function to construct a layer, the <code>geom</code> argument can be used to override the default coupling between stats and geoms. The <code>geom</code> argument accepts the following:</p> <ul style="list-style-type: none"> <li>• A <code>Geom</code> ggproto subclass, for example <code>GeomPoint</code>.</li> <li>• A string naming the geom. To give the geom as a string, strip the function name of the <code>geom_</code> prefix. For example, to use <code>geom_point()</code>, give the geom as "point".</li> <li>• For more information and other ways to specify the geom, see the <a href="#">layer geom</a> documentation.</li> </ul>
probs	Quantiles to use for the five number summary.
na.rm	If <code>FALSE</code> (the default), removes missing values with a warning. If <code>TRUE</code> silently removes missing values.
position	<p>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 <code>position</code> argument accepts the following:</p> <ul style="list-style-type: none"> <li>• The result of calling a position function, such as <code>position_jitter()</code>. This method allows for passing extra arguments to the position.</li> <li>• A string naming the position adjustment. To give the position as a string, strip the function name of the <code>position_</code> prefix. For example, to use <code>position_jitter()</code>, give the position as "jitter".</li> <li>• For more information and other ways to specify the position, see the <a href="#">layer position</a> documentation.</li> </ul>
show.legend	logical. Should this layer be included in the legends? <code>NA</code> , the default, includes if any aesthetics are mapped. <code>FALSE</code> never includes, and <code>TRUE</code> always includes. It can also be a named logical vector to finely select the aesthetics to display. To include legend keys for all levels, even when no data exists, use <code>TRUE</code> . If <code>NA</code> , all levels are shown in legend, but unobserved levels are omitted.
inherit.aes	If <code>FALSE</code> , overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. <a href="#">annotation_borders()</a> .

... 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:

<code>width</code>	width of boxplot
<code>min</code>	minimum
<code>lower</code>	lower hinge, 25% quantile
<code>middle</code>	median, 50% quantile
<code>upper</code>	upper hinge, 75% quantile
<code>max</code>	maximum

**See Also**

[stat\\_boxplot\(\)](#)

---

<code>tableau_color_pal</code>	<i>Tableau Color Palettes (discrete)</i>
--------------------------------	--

---

**Description**

Color palettes used in [Tableau](#).

**Usage**

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

**Arguments**

palette	Palette name. See Details for available palettes.
type	Type of palette. One of "regular", "ordered-diverging", or "ordered-sequential".
direction	If 1, the default, then use the original order of colors. If -1, then 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", "Red-Blue-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"

**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 colour 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	<i>Tableau colour gradient palettes (continuous)</i>
----------------------	--

---

Description

Gradient color palettes using the diverging and sequential continous 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.  "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      Palette type, either "ordered-sequential" or "ordered-diverging".
...       Arguments passed to tableau_gradient_pal.
```

See Also

Other colour 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	<i>Tableau Shape Palettes (discrete)</i>
-------------------	--

---

Description

Shape palettes used by **Tableau**.

Usage

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

Arguments

palette            Palette name.

Details

Not all shape palettes in Tableau are supported. Additionally, these palettes are not exact, and use the best unicode character for the shape palette.

Since these palettes use unicode characters, their look may depend on the font being used, and not all characters may be available.

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

**See Also**

Other shape tableau: [scale\\_shape\\_tableau\(\)](#)

**Examples**

```
## Not run:
# need to set a font containing these values
show_shapes(tableau_shape_pal()(5))

## End(Not run)
```

---

theme_base	<i>Theme Base</i>
------------	-------------------

---

**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

**See Also**

Other themes: [theme\\_clean\(\)](#), [theme\\_foundation\(\)](#), [theme\\_igray\(\)](#), [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
par(fg = "blue", bg = "gray", col.lab = "red", font.lab = 3)
p + theme_base()
```

---

`theme_calc`*Theme Calc*

---

**Description**

Theme similar to the default settings of LibreOffice Calc charts.

**Usage**

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

**Arguments**

<code>base_size</code>	base font size, given in pts.
<code>base_family</code>	base font family

**Examples**

```
library("ggplot2")

ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, colour = factor(gear))) +
  facet_wrap(~am) +
  theme_calc() +
  scale_color_calc()
## Not run:
ggplot(mtcars) +
  geom_point(aes(x = wt, y = mpg, shape = factor(gear))) +
  facet_wrap(~am) +
  theme_calc() +
  scale_shape_calc()

## End(Not run)
```

---

`theme_clean`*Clean ggplot theme*

---

**Description**

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

**Usage**

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

**Arguments**

base\_size      Base font size.  
base\_family    Base font family.

**Author(s)**

Konrad Zdeb <name.surname@me.com>

**See Also**

Other themes: [theme\\_base\(\)](#), [theme\\_foundation\(\)](#), [theme\\_igray\(\)](#), [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	<i>ggplot color theme based on the Economist</i>
-----------------	--

---

**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	logical Horizontal axis lines?
dkpanel	logical Darker background for panel region?
gray_bg	logical If TRUE, use gray background, else use white background.

## Details

theme\_economist implements the standard bluish-gray background theme in the print *The Economist* and [economist.com](http://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 x 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 [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	logical. Horizontal axis lines?

## Value

An object of class [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

**Value**

An object of class `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

## 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](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	<i>Theme inspired by FiveThirtyEight plots</i>
-----------------------	--

---

### 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

### 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	<i>Foundation Theme</i>
------------------	-------------------------

---

### 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 `theme_gray()` or `theme_bw()`, because those themes define elements deep in the hierarchy.

### Usage

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

### Arguments

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

Details

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

See Also

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

---

theme_gdocs	<i>Theme with Google Docs Chart defaults</i>
-------------	--

---

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

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	<i>Highcharts Theme</i>
----------	-------------------------

---

Description

Theme based on **Highcharts** plots.

**Usage**

```
theme_hc(  
  base_size = 12,  
  base_family = "sans",  
  style = c("default", "darkunica"),  
  bgcolor = NULL  
)
```

**Arguments**

base_size	base font size, given in pts.
base_family	base font family
style	The Highcharts theme to use 'default', 'darkunica'.
bgcolor	Deprecated

**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(bgcolor = "darkunica") + scale_colour_hc("darkunica")  
  
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(bgcolor = "darkunica") +
  scale_fill_hc("darkunica")

```

---

theme_igray	<i>Inverse gray theme</i>
-------------	---------------------------

---

## 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 in the [theme\\_gray\(\)](#), a white panel and a light gray area around it. This keeps a white background for the color scales like [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 [theme\\_gray\(\)](#). This is similar to the style of plots in Stata and Tableau.

## See Also

[theme\\_gray\(\)](#), [theme\\_bw\(\)](#)

Other themes: [theme\\_base\(\)](#), [theme\\_clean\(\)](#), [theme\\_foundation\(\)](#), [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 `geom_map()`.

### Usage

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

### Arguments

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

### Examples

```
library("maps")
library("ggplot2")

us <- fortify(map_data("state"), region = "region")
gg <- ggplot() +
  geom_map(
    data = us,
    map = us,
    aes(x = long, y = lat, map_id = region, group = group),
    fill = "white",
    color = "black",
    size = 0.25
  ) +
  coord_map("albers", lat0 = 39, lat1 = 45) +
  theme_map()
gg
```

---

 theme\_pander

*A ggplot theme originated from the pander package*


---

### Description

The **pander** 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.msgrs` 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	suppress the white space around the plot (boolean)
ff	font family, like sans. Deprecated: use base_family instead.
fc	font color (name or hexa code)
fs	font size (integer). Deprecated: use base_size instead.
gM	major grid (boolean)
gm	minor grid (boolean)
gc	grid color (name or hexa code)
gl	grid line type (lty)
boxes	to render a border around the plot or not
bc	background color (name or hexa code)
pc	panel background color (name or hexa code)
lp	legend position
axis	axis angle as defined in par(las)

**Examples**

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

```

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

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	<i>Theme which uses the current 'base' graphics parameter values from <a href="#">par()</a>. Not all <a href="#">par()</a> parameters, are supported, and not all are relevant to <b>ggplot2</b> themes.</i>
-----------	--

---

## Description

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

## 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

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.

## See Also

Other themes: [theme\\_base\(\)](#), [theme\\_clean\(\)](#), [theme\\_foundation\(\)](#), [theme\\_igray\(\)](#), [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
par(font = 2, col.lab = "red", fg = "white", bg = "black")
p + theme_par()
```

---

theme_solarized	<i>ggplot color themes based on the Solarized palette</i>
-----------------	---

---

**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	logical. Light or 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 [theme\\_bw\(\)](#), while theme\_solarized\_2() is similar to [theme\\_gray\(\)](#).

**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	<i>Theme with nothing other than a background color</i>
-------------	---

---

## Description

Theme that removes all non-geom elements (lines, text, etc), This theme is when only the geometric objects are desired.

## Usage

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

## Arguments

base_size	Base font size.
base_family	Ignored, kept for consistency with <a href="#">theme()</a> .
fill	Background color of the plot.

## See Also

Other themes: [theme\\_base\(\)](#), [theme\\_clean\(\)](#), [theme\\_foundation\(\)](#), [theme\\_igray\(\)](#), [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 = "s2color")
```

## Arguments

base_size	base font size, given in pts.
base_family	base font family
scheme	One of "s2color", "s2mono", "s1color", "s1rcolor", or "s1mono", "s2manual", "s1manual", or "sj"

## 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.

## 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"
  )

# s2color
p + theme_stata() + scale_colour_stata("s2color")
# s2mono
p + theme_stata(scheme = "s2mono") + scale_colour_stata("mono")
# s1color
```

```
p + theme_stata(scheme = "s2color") + 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 [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	logical Show axis ticks?

## 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.

## 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	The background color of plot. One of 'brown', 'gray', 'green', 'blue'.
base_family	base font family
title_family	Plot title font family.

## Details

This theme should be used with [scale\\_color\\_wsj\(\)](#).

## 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	<i>Shape palette from Tremmel (1995) (discrete)</i>
-------------------	---

---

## 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.

## 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.

## 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\(\)](#)

---

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	character The color palette to use: . "rgby", "red_green", "black_green", "dem_rep", "colors6"
---------	--

### Palettes

The following palettes are defined,

**rgby** Red/Green/Blue/Yellow theme.

**red\_green** Green/red two-color scale for good/bad.

**green\_black** 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 colour wsj: [scale\\_colour\\_wsj\(\)](#)

# Index

- \* **colour calc**
  - calc\_pal, [6](#)
  - scale\_fill\_calc, [46](#)
- \* **colour economist**
  - economist\_pal, [13](#)
  - scale\_colour\_economist, [29](#)
- \* **colour excel**
  - excel\_new\_pal, [14](#)
  - excel\_pal, [14](#)
  - scale\_colour\_excel\_new, [30](#)
  - scale\_fill\_excel, [48](#)
- \* **colour few**
  - few\_pal, [16](#)
  - scale\_colour\_few, [32](#)
- \* **colour fivethirtyeight**
  - fivethirtyeight\_pal, [17](#)
  - scale\_colour\_fivethirtyeight, [33](#)
- \* **colour gdocs**
  - gdocs\_pal, [18](#)
  - scale\_fill\_gdocs, [49](#)
- \* **colour hc**
  - hc\_pal, [25](#)
  - scale\_colour\_hc, [39](#)
- \* **colour pander**
  - palette\_pander, [25](#)
  - scale\_color\_pander, [27](#)
- \* **colour ptol**
  - ptol\_pal, [26](#)
  - scale\_colour\_ptol, [40](#)
- \* **colour scales**
  - scale\_fill\_solarized, [51](#)
- \* **colour stata**
  - scale\_colour\_stata, [42](#)
- \* **colour tableau**
  - scale\_colour\_gradient2\_tableau, [35](#)
  - scale\_colour\_gradient\_tableau, [37](#)
  - scale\_colour\_tableau, [43](#)
  - tableau\_color\_pal, [70](#)
  - tableau\_gradient\_pal, [72](#)
- \* **colour wsj**
  - scale\_colour\_wsj, [45](#)
  - wsj\_pal, [95](#)
- \* **colour**
  - colorblind\_pal, [11](#)
- \* **datasets**
  - canva\_palettes, [8](#)
  - geom\_rangeframe, [18](#)
  - geom\_tufteboxplot, [21](#)
  - ggthemes\_data, [24](#)
  - stat\_fivenumber, [68](#)
- \* **geom tufte**
  - geom\_rangeframe, [18](#)
  - geom\_tufteboxplot, [21](#)
- \* **linetype stata**
  - scale\_linetype\_stata, [52](#)
  - stata\_linetype\_pal, [67](#)
- \* **shape stata**
  - scale\_shape\_stata, [59](#)
- \* **shape tableau**
  - scale\_shape\_tableau, [60](#)
  - tableau\_shape\_pal, [73](#)
- \* **shapes calc**
  - calc\_shape\_pal, [6](#)
  - scale\_shape\_calc, [54](#)
- \* **shapes stata**
  - stata\_shape\_pal, [68](#)
- \* **shapes**
  - circlefill\_shape\_pal, [9](#)
  - cleveland\_shape\_pal, [10](#)
  - scale\_shape\_circlefill, [55](#)
  - scale\_shape\_cleveland, [57](#)
  - scale\_shape\_tremmel, [62](#)
  - tremmel\_shape\_pal, [94](#)
- \* **solarized colour**
  - scale\_fill\_solarized, [51](#)
  - solarized\_pal, [66](#)
- \* **stata colour**
  - stata\_pal, [67](#)



- \* **themes calc**
    - theme\_calc, 75
  - \* **themes economist**
    - theme\_economist, 76
  - \* **themes excel**
    - theme\_excel, 78
    - theme\_excel\_new, 79
  - \* **themes few**
    - theme\_few, 80
  - \* **themes fivethirtyeight**
    - theme\_fivethirtyeight, 81
  - \* **themes gdocs**
    - theme\_gdocs, 82
  - \* **themes hc**
    - theme\_hc, 82
  - \* **themes solarized**
    - theme\_solarized, 89
  - \* **themes stata**
    - theme\_stata, 91
  - \* **themes tufte**
    - theme\_tufte, 92
  - \* **themes wsj**
    - theme\_wsj, 93
  - \* **themes**
    - theme\_base, 74
    - theme\_clean, 75
    - theme\_foundation, 81
    - theme\_igray, 85
    - theme\_par, 88
    - theme\_solid, 90
- aes(), 19, 22, 69  
 annotation\_borders(), 20, 23, 69  
 bank\_slopes, 4  
 banking, 5  
 calc\_pal, 6, 47  
 calc\_shape\_pal, 6, 54, 55  
 canva\_pal, 7  
 canva\_palettes, 7, 8, 28  
 circlefill\_shape\_pal, 9, 10, 57, 58, 63, 94  
 cleveland\_shape\_pal, 9, 10, 57, 58, 63, 94  
 colorblind\_pal, 11  
 colourblind\_pal (colorblind\_pal), 11  
 dichromat\_pal, 12  
 discrete\_scale, 11, 27–29, 31, 32, 34, 39, 40, 42, 44–46, 48, 50, 51, 53, 54, 56, 57, 59, 61, 62  
 economist\_pal, 13, 30  
 excel\_new\_pal, 14, 15, 32, 49  
 excel\_pal, 14, 14, 32, 49  
 extended\_range\_breaks  
     (extended\_range\_breaks\_), 15  
 extended\_range\_breaks\_, 15  
 few\_pal, 16, 32, 33  
 few\_shape\_pal, 17  
 fivethirtyeight\_pal, 17, 35  
 format, 66  
 fortify(), 19, 22, 69  
 gdocs\_pal, 18, 51  
 geom\_boxplot, 24  
 geom\_map, 86  
 geom\_rangeframe, 18, 24, 92  
 geom\_rug, 92  
 geom\_tufteboxplot, 21, 21  
 GeomRangeFrame (geom\_rangeframe), 18  
 GeomTufteboxplot (geom\_tufteboxplot), 21  
 ggplot(), 19, 22, 69  
 ggthemes\_data, 24  
 guides(), 12, 28, 30, 31, 33, 35, 40, 41, 43, 46, 47, 49, 50, 52  
 hc\_pal, 25, 39, 40  
 key\_glyphs, 20, 23, 70  
 lambda, 11, 12, 27–35, 39–43, 45–63  
 layer geom, 69  
 layer position, 20, 22, 69  
 layer stat, 19  
 layer(), 20, 23, 70  
 palette\_pander, 25, 28  
 par, 64, 65, 88  
 ptol\_pal, 26, 40, 41  
 scale\_color\_calc (scale\_fill\_calc), 46  
 scale\_color\_canva (scale\_colour\_canva), 28  
 scale\_color\_colorblind  
     (colorblind\_pal), 11  
 scale\_color\_continuous\_tableau  
     (scale\_colour\_gradient\_tableau), 37  
 scale\_color\_economist, 77

scale\_color\_economist  
     (scale\_colour\_economist), 29  
 scale\_color\_excel (scale\_fill\_excel), 48  
 scale\_color\_excel\_new  
     (scale\_colour\_excel\_new), 30  
 scale\_color\_few (scale\_colour\_few), 32  
 scale\_color\_fivethirtyeight  
     (scale\_colour\_fivethirtyeight),  
     33  
 scale\_color\_gdocs (scale\_fill\_gdocs), 49  
 scale\_color\_gradient2\_tableau  
     (scale\_colour\_gradient2\_tableau),  
     35  
 scale\_color\_gradient\_tableau  
     (scale\_colour\_gradient\_tableau),  
     37  
 scale\_color\_hc (scale\_colour\_hc), 39  
 scale\_color\_pander, 25, 27  
 scale\_color\_ptol (scale\_colour\_ptol), 40  
 scale\_color\_solarized  
     (scale\_fill\_solarized), 51  
 scale\_color\_stata (scale\_colour\_stata),  
     42  
 scale\_color\_tableau, 12  
 scale\_color\_tableau  
     (scale\_colour\_tableau), 43  
 scale\_color\_wsj, 93  
 scale\_color\_wsj (scale\_colour\_wsj), 45  
 scale\_colour\_calc (scale\_fill\_calc), 46  
 scale\_colour\_canva, 28  
 scale\_colour\_colorblind  
     (colorblind\_pal), 11  
 scale\_colour\_colourblind  
     (colorblind\_pal), 11  
 scale\_colour\_economist, 13, 29  
 scale\_colour\_excel, 78  
 scale\_colour\_excel (scale\_fill\_excel),  
     48  
 scale\_colour\_excel\_new, 14, 15, 30, 49  
 scale\_colour\_few, 17, 32  
 scale\_colour\_fivethirtyeight, 18, 33  
 scale\_colour\_gdocs (scale\_fill\_gdocs),  
     49  
 scale\_colour\_gradient2\_tableau, 35, 38,  
     44, 72, 73  
 scale\_colour\_gradient2\_tableau(), 37,  
     43  
 scale\_colour\_gradient\_tableau, 36, 37,  
     44, 72, 73  
 scale\_colour\_gradient\_tableau(), 35, 43  
 scale\_colour\_hc, 25, 39  
 scale\_colour\_pander  
     (scale\_color\_pander), 27  
 scale\_colour\_ptol, 26, 40  
 scale\_colour\_solarized  
     (scale\_fill\_solarized), 51  
 scale\_colour\_stata, 42  
 scale\_colour\_tableau, 36, 38, 43, 72, 73  
 scale\_colour\_tableau(), 35, 37  
 scale\_colour\_wsj, 45, 95  
 scale\_fill\_calc, 6, 46  
 scale\_fill\_canva (scale\_colour\_canva),  
     28  
 scale\_fill\_colorblind (colorblind\_pal),  
     11  
 scale\_fill\_colourblind  
     (colorblind\_pal), 11  
 scale\_fill\_continuous\_tableau  
     (scale\_colour\_gradient\_tableau),  
     37  
 scale\_fill\_economist  
     (scale\_colour\_economist), 29  
 scale\_fill\_excel, 14, 15, 32, 48  
 scale\_fill\_excel\_new  
     (scale\_colour\_excel\_new), 30  
 scale\_fill\_few (scale\_colour\_few), 32  
 scale\_fill\_fivethirtyeight  
     (scale\_colour\_fivethirtyeight),  
     33  
 scale\_fill\_gdocs, 18, 49  
 scale\_fill\_gradient2\_tableau  
     (scale\_colour\_gradient2\_tableau),  
     35  
 scale\_fill\_gradient\_tableau  
     (scale\_colour\_gradient\_tableau),  
     37  
 scale\_fill\_hc (scale\_colour\_hc), 39  
 scale\_fill\_pander (scale\_color\_pander),  
     27  
 scale\_fill\_ptol (scale\_colour\_ptol), 40  
 scale\_fill\_solarized, 51, 66  
 scale\_fill\_stata (scale\_colour\_stata),  
     42  
 scale\_fill\_tableau  
     (scale\_colour\_tableau), 43  
 scale\_fill\_wsj (scale\_colour\_wsj), 45

scale\_linetype\_stata, 52, 67  
scale\_shape\_calc, 6, 54  
scale\_shape\_circlefill, 9, 10, 55, 58, 63, 94  
scale\_shape\_cleveland, 9, 10, 57, 57, 63, 94  
scale\_shape\_few, 58, 59  
scale\_shape\_stata, 59, 68  
scale\_shape\_tableau, 60, 74  
scale\_shape\_tremmel, 9, 10, 57, 58, 62, 94  
show\_col, 64, 65  
show\_linetypes, 64, 64, 65  
show\_shapes, 65  
smart\_digits, 65  
smart\_digits\_format (smart\_digits), 65  
solarized\_pal, 51, 52, 66  
stat\_boxplot, 22, 70  
stat\_fivenumber, 22, 68  
stata\_linetype\_pal, 52, 54, 67  
stata\_pal, 42, 67  
stata\_shape\_pal, 59, 68  
StatFivenumber (stat\_fivenumber), 68  
  
tableau\_color\_pal, 36, 38, 44, 70, 73  
tableau\_color\_pal(), 44, 72  
tableau\_div\_gradient\_pal  
    (tableau\_gradient\_pal), 72  
tableau\_gradient\_pal, 36, 38, 44, 72, 72  
tableau\_seq\_gradient\_pal  
    (tableau\_gradient\_pal), 72  
tableau\_shape\_pal, 60, 62, 73  
theme, 77–79, 90  
theme\_base, 74, 76, 82, 85, 88, 90  
theme\_bw, 81, 85, 89  
theme\_calc, 47, 55, 75  
theme\_clean, 74, 75, 82, 85, 88, 90  
theme\_economist, 30, 76  
theme\_economist\_white  
    (theme\_economist), 76  
theme\_excel, 78, 79  
theme\_excel\_new, 78, 79  
theme\_few, 80  
theme\_fivethirtyeight, 35, 81  
theme\_foundation, 74, 76, 81, 85, 88, 90  
theme\_gdocs, 51, 82  
theme\_gray, 81, 82, 85, 89  
theme\_hc, 39, 82  
theme\_igray, 74, 76, 82, 85, 88, 90  
theme\_map, 86  
theme\_pander, 28, 86  
  
theme\_par, 74, 76, 82, 85, 88, 90  
theme\_solarized, 89  
theme\_solarized\_2 (theme\_solarized), 89  
theme\_solid, 74, 76, 82, 85, 88, 90  
theme\_stata, 91  
theme\_tufte, 92  
theme\_wsj, 45, 93  
tremmel\_shape\_pal, 9, 10, 57, 58, 63, 94  
wsj\_pal, 45, 46, 95