

Package ‘khroma’

April 23, 2021

Title Colour Schemes for Scientific Data Visualization

Version 1.5.0

Maintainer Nicolas Frerebeau <nicolas.frerebeau@u-bordeaux-montaigne.fr>

Description Colour schemes ready for each type of data (qualitative, diverging or sequential), with colours that are distinct for all people, including colour-blind readers. This package provides an implementation of Paul Tol (2018) and Fabio Crameri (2018) <doi:10.5194/gmd-11-2541-2018> colour schemes for use with 'graphics' or 'ggplot2'. It provides tools to simulate colour-blindness and to test how well the colours of any palette are identifiable. Several scientific thematic schemes (geologic timescale, land cover, FAO soils, etc.) are also implemented.

License GPL-3

URL <https://khroma.tesselle.org>, <https://github.com/tesselle/khroma>

BugReports <https://github.com/tesselle/khroma/issues>

Depends R (>= 3.3)

Imports ggplot2, grDevices, grid, scales, spacesXYZ

Suggests covr, crayon, fansi, knitr, rmarkdown, testthat (>= 3.0.0), vdiff

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.1.1

Collate 'anomalize.R' 'colour.R' 'compare.R' 'convert.R' 'ggplot2.R' 'khroma-package.R' 'plot.R' 'scale_colour_crameri.R' 'scale_colour_okabeito.R' 'scale_colour_science.R' 'scale_colour_tol.R'

NeedsCompilation no

Author Nicolas Frerebeau [aut, cre] (<<https://orcid.org/0000-0001-5759-4944>>),
Brice Lebrun [ctb] (<<https://orcid.org/0000-0001-7503-8685>>),
Vincent Arel-Bundock [ctb] (<<https://orcid.org/0000-0003-2042-7063>>)

Repository CRAN

Date/Publication 2021-04-23 04:30:07 UTC

R topics documented:

colour	2
compare	5
convert	6
plot	8
scale_colour_land	9
scale_colour_soil	10
scale_colour_stratigraphy	12
scale_crameri_cyclic	13
scale_crameri_diverging	17
scale_crameri_mutlisequential	24
scale_crameri_sequential	27
scale_okabeito_discrete	38
scale_tol_discrete	39
scale_tol_diverging	42
scale_tol_sequential	45

Index **49**

colour *Color Palette*

Description

Provides qualitative, diverging and sequential color schemes.

Usage

```
colour(palette, reverse = FALSE, names = TRUE, lang = "en", force = FALSE, ...)
```

```
color(palette, reverse = FALSE, names = TRUE, lang = "en", force = FALSE, ...)
```

Arguments

palette	A character string giving the name of the palette to be used (see below).
reverse	A logical scalar: should the resulting vector of colors should be reversed?
names	A logical scalar: should the names of the colors should be kept in the resulting vector?
lang	A character string specifying the language for the color names. It must be one of "en" (english, the default) or "fr" (french).
force	A logical scalar. If TRUE, forces the color scheme to be interpolated. It should not be used routinely with qualitative color schemes, as they are designed to be used as is to remain colorblind-safe.
...	Further arguments passed to colorRampPalette .

Value

A palette function with the following attributes, that when called with a single integer argument (the number of levels) returns a (named) vector of colors.

palette A `character` string giving the name of the color scheme.

type A `character` string giving the corresponding data type. One of "qualitative", "diverging" or "sequential".

interpolate A `logical` scalar: can the color palette be interpolated?

missing A `character` string giving the the hexadecimal representation of the color that should be used for NA values.

max An `integer` giving the maximum number of color values. Only relevant for non-interpolated color schemes.

For colour schemes that can be interpolated (diverging and sequential data), the colour range can be limited with an additional argument. `range` allows to remove a fraction of the colour domain (before being interpolated; see examples).

Paul Tol's Color Schemes

The following palettes are available. The maximum number of supported colors is in brackets, this value is only relevant for the qualitative color schemes (divergent and sequential schemes are linearly interpolated).

Qualitative data bright (7), contrast (3), vibrant (7), muted (9), pale (6), dark (6), light (9).

Diverging data sunset (11), BuRd (9), PRGn (9).

Sequential data YlOrBr (9), iridescent (23), discrete rainbow (23), smooth rainbow (34).

Qualitative color schemes

According to Paul Tol's technical note, the bright, contrast, vibrant and muted color schemes are colorblind safe.

The light color scheme is reasonably distinct for both normal or colorblind vision and is intended to fill labeled cells.

The pale and dark schemes are not very distinct in either normal or colorblind vision and should be used as a text background or to highlight a cell in a table.

Refer to the original document for details about the recommended uses (see references).

Rainbow color scheme

As a general rule, ordered data should not be represented using a rainbow scheme. There are three main arguments against such use (Tol 2018):

- The spectral order of visible light carries no inherent magnitude message.
- Some bands of almost constant hue with sharp transitions between them, can be perceived as jumps in the data.
- Colour-blind people have difficulty distinguishing some colours of the rainbow.

If such use cannot be avoided, Paul Tol's technical note provides two colour schemes that are reasonably clear in colour-blind vision. To remain colour-blind safe, these two schemes must comply with the following conditions:

discrete rainbow This scheme must not be interpolated.

smooth rainbow This scheme does not have to be used over the full range.

Okabe and Ito Colour Scheme

The following (qualitative) colour scheme is available:

okabe ito Up to 8 colours.

Scientific Colour Schemes

The following (qualitative) color schemes are available:

stratigraphy International Chronostratigraphic Chart (175 colours).

land AVHRR Global Land Cover Classification (14 colours).

soil FAO Reference Soil Groups (24 colours).

Author(s)

N. Frerebeau

References

Jones, A., Montanarella, L. & Jones, R. (Ed.) (2005). *Soil atlas of Europe*. Luxembourg: European Commission, Office for Official Publications of the European Communities. 128 pp. ISBN: 92-894-8120-X.

Okabe, M. & Ito, K. (2008). *Color Universal Design (CUD): How to Make Figures and Presentations That Are Friendly to Colorblind People*. URL: <https://jfly.uni-koeln.de/color/>.

Tol, P. (2018). *Colour Schemes*. SRON. Technical Note No. SRON/EPS/TN/09-002, issue 3.1. URL: <https://personal.sron.nl/~pault/data/colourschemes.pdf>

Commission for the Geological Map of the World

Examples

```
## Okabe and Ito colour scheme
colour("okabe ito")(8)
plot_scheme(colour("okabe ito")(8))
```

```
## Paul Tol's colour schemes
### Qualitative data
plot_scheme(colour("bright")(7))
plot_scheme(colour("contrast")(3))
plot_scheme(colour("vibrant")(7))
plot_scheme(colour("pale")(6))
plot_scheme(colour("dark")(6))
plot_scheme(colour("muted")(9))
```

```

plot_scheme(colour("light")(9))
### Diverging data
plot_scheme(colour("sunset")(11))
plot_scheme(colour("BuRd")(9))
plot_scheme(colour("PRGn")(9))
### Sequential data
plot_scheme(colour("YlOrBr")(9))
plot_scheme(colour("iridescent")(23))
plot_scheme(colour("discrete rainbow")(14))
plot_scheme(colour("discrete rainbow")(23))
plot_scheme(colour("smooth rainbow")(34))

## Scientific colour schemes
### Geologic timescale
plot_scheme(colour("stratigraphy")(175))
### AVHRR global land cover classification
plot_scheme(colour("land")(14))
### FAO soil reference groups
plot_scheme(colour("soil")(24))

## Adjust colour levels
PRGn <- colour("PRGn")
plot_scheme(PRGn(9, range = c(0.5, 1)))

```

compare

Colour Difference

Description

Computes CIELAB distance metric.

Usage

```
compare(x, metric = 2000, diag = FALSE, upper = FALSE)
```

Arguments

x	A character vector of colors.
metric	An integer value giving the year the metric was recommended by the CIE. It must be one of "1976", "1994", or "2000" (default; see spacesXYZ::DeltaE()).
diag	A logical scalar: should the diagonal of the distance matrix be printed?
upper	A logical scalar: should the upper triangle of the distance matrix should be printed?

Value

A [distance matrix](#).

Author(s)

N. Frerebeau

See Also

Other diagnostic tools: [convert\(\)](#), [plot\(\)](#)

Examples

```
# Trichromat
pal <- colour("bright")

(deltaE <- compare(pal(5)))
summary(deltaE)

# Deuteranopia
deu <- convert(pal, mode = "deuteranopia")
compare(deu(5))

# Protanopia
pro <- convert(pal, mode = "protanopia")
compare(pro(5))

# Tritanopia
tri <- convert(pal, mode = "tritanopia")
compare(tri(5))

# Achromatopsia
ach <- convert(pal, mode = "achromatopsia")
compare(ach(5))
```

convert

Simulate Colour-Blindness

Description

Simulate Colour-Blindness

Usage

```
convert(x, mode)
```

Arguments

x	A palette function that when called with a single integer argument (the number of levels) returns a vector of colors (see colour()).
mode	A character string giving the colorblind vision to be used. It must be one of "deuteranopia", "protanopia", "tritanopia" or "achromatopsia". Any unambiguous substring can be given.

Value

A palette `function` that returns a vector of anomalized colours. All the attributes of the initial palette function are inherited, with a supplementary attribute "mode" giving the corresponding colour-blind vision.

Author(s)

N. Frerebeau

References

Brettel, H., Viénot, F. and Mollon, J. D. (1997). Computerized Simulation of Color Appearance for Dichromats. *Journal of the Optical Society of America A*, 14(10), p. 2647-2655. doi: [10.1364/JOSAA.14.002647](https://doi.org/10.1364/JOSAA.14.002647).

Tol, P. (2018). *Colour Schemes*. SRON. Technical Note No. SRON/EPS/TN/09-002, issue 3.1. URL: <https://personal.sron.nl/~pault/data/colourschemes.pdf>

Viénot, F., Brettel, H. and Mollon, J. D. (1999). Digital Video Colourmaps for Checking the Legibility of Displays by Dichromats. *Color Research & Application*, 24(4), p. 243-52. doi: [10.1002/\(SICI\)15206378\(199908\)24:4<243::AIDCOL5>3.0.CO;23](https://doi.org/10.1002/(SICI)15206378(199908)24:4<243::AIDCOL5>3.0.CO;23).

See Also

Other diagnostic tools: `compare()`, `plot()`

Examples

```
# Trichromat
pal <- colour("bright")
plot_scheme(pal(7))

# Deuteranopia
deu <- convert(pal, mode = "deuteranopia")
plot_scheme(deu(7))

# Protanopia
pro <- convert(pal, mode = "protanopia")
plot_scheme(pro(7))

# Tritanopia
tri <- convert(pal, mode = "tritanopia")
plot_scheme(tri(7))

# Achromatopsia
ach <- convert(pal, mode = "achromatopsia")
plot_scheme(ach(7))
```

plot

Plot Color Scheme

Description

- `plot()` allows to quickly display a color scheme returned by `colour()`.
- `plot_scheme()` shows colors in a plot.
- `plot_map()` produces a diagnostic map for a given color scheme.
- `plot_scheme_colorblind()` shows colors in a plot with different types of simulated color blindness.

Usage

```
## S3 method for class 'colour_scheme'  
plot(x, ...)  
  
plot_scheme(x, colours = FALSE, names = FALSE, size = 1)  
  
plot_map(x)  
  
plot_scheme_colourblind(x)  
  
plot_scheme_colorblind(x)
```

Arguments

<code>x</code>	A character vector of colors.
<code>...</code>	Currently not used.
<code>colours</code>	A logical scalar: should the hexadecimal representation of the colors be displayed?
<code>names</code>	A logical scalar: should the name of the colors be displayed?
<code>size</code>	A numeric value giving the amount by which plotting text should be magnified relative to the default. Works the same as <code>cex</code> parameter of graphics::par() .

Author(s)

N. Frerebeau, V. Arel-Bundock

See Also

Other diagnostic tools: [compare\(\)](#), [convert\(\)](#)

Examples

```

plot(colour("bright")(7))
plot(colour("smooth rainbow")(256))

## Plot colour schemes
plot_scheme(colour("bright")(7))
plot_scheme(colour("sunset")(11))
plot_scheme(colour("YlOrBr")(9))
plot_scheme(colour("discrete rainbow")(14))

## Plot diagnostic maps
plot_map(colour("bright")(7))
plot_map(colour("sunset")(11))
plot_map(colour("YlOrBr")(9))
plot_map(colour("discrete rainbow")(14))

## Plot simulated color blindness
plot_scheme_colorblind(colour("bright")(7))

```

scale_colour_land *AVHRR Global Land Cover Classification Colour Scheme for ggplot2*

Description

Provides the AVHRR Global Land Cover classification as modified by Paul Tol (colorblind safe).

Usage

```

scale_colour_land(..., lang = "en", aesthetics = "colour")

scale_color_land(..., lang = "en", aesthetics = "colour")

scale_fill_land(..., lang = "en", aesthetics = "fill")

```

Arguments

...	Arguments passed on to <code>ggplot2::discrete_scale()</code> .
lang	A character string specifying the language for the color names (see details). It must be one of "en" (english, the default) or "fr" (french).
aesthetics	A character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

Values will be matched based on the soil names.

Value

A [discrete](#) scale.

Author(s)

N. Frerebeau

References

Tol, P. (2018). *Colour Schemes*. SRON. Technical Note No. SRON/EPS/TN/09-002, issue 3.1.
 URL: <https://personal.sron.nl/~pault/data/colourschemes.pdf>

See Also

Other themed colour schemes: [scale_colour_soil\(\)](#), [scale_colour_stratigraphy\(\)](#)

Other qualitative colour schemes: [scale_colour_soil\(\)](#), [scale_colour_stratigraphy\(\)](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#)

Examples

```
library(ggplot2)

land <- data.frame(
  name = c(
    "water", "evergreen needleleaf forest", "deciduous needleleaf forest",
    "mixed forest", "evergreen broadleaf forest", "deciduous broadleaf forest",
    "woodland", "wooded grassland", "grassland", "cropland", "closed shrubland",
    "open shrubland", "bare ground", "urban and built"
  )
)

ggplot2::ggplot(land, ggplot2::aes(fill = name)) +
  ggplot2::geom_rect(aes(xmin = rep(0, 14), xmax = rep(1, 14),
                        ymin = 1:14, ymax = 1:14+1)) +
  ggplot2::scale_y_reverse() +
  scale_fill_land(name = "land")
```

 scale_colour_soil

FAO Soil Reference Groups Color Scheme for ggplot2

Description

Provides the FAO Soil Reference Groups colour scheme.

Usage

```
scale_colour_soil(..., lang = "en", aesthetics = "colour")
```

```
scale_color_soil(..., lang = "en", aesthetics = "colour")
```

```
scale_fill_soil(..., lang = "en", aesthetics = "fill")
```

Arguments

...	Arguments passed on to <code>ggplot2::discrete_scale()</code> .
lang	A character string specifying the language for the color names (see details). It must be one of "en" (english, the default) or "fr" (french).
aesthetics	A character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

Values will be matched based on the soil names.

Value

A [discrete](#) scale.

Author(s)

N. Frerebeau

References

Jones, A., Montanarella, L. & Jones, R. (Ed.) (2005). *Soil atlas of Europe*. Luxembourg: European Commission, Office for Official Publications of the European Communities. 128 pp. ISBN: 92-894-8120-X.

See Also

Other themed colour schemes: [scale_colour_land\(\)](#), [scale_colour_stratigraphy\(\)](#)

Other qualitative colour schemes: [scale_colour_land\(\)](#), [scale_colour_stratigraphy\(\)](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#)

Examples

```
library(ggplot2)

soil <- data.frame(
  name = c(
    "Acrisol", "Albeluvisol", "Andosol", "Anthrosol", "Arenosol", "Calcisol",
    "Cambisol", "Chernozem", "Cryosol", "Fluvisol", "Kastanozem", "Gleysol",
    "Gypsisol", "Histosol", "Leptosol", "Luvisol", "Phaeozem", "Planosol",
    "Podzol", "Regosol", "Solonchak", "Solonetz", "Umbrisol", "Vertisol"
  )
)

ggplot2::ggplot(soil, ggplot2::aes(fill = name)) +
  ggplot2::geom_rect(aes(xmin = rep(0, 24), xmax = rep(1, 24),
    ymin = 1:24, ymax = 1:24+1)) +
  ggplot2::scale_y_reverse() +
  scale_fill_soil(name = "Soil")
```

scale_colour_stratigraphy

*Geologic Timescale Color Scheme for **ggplot2***

Description

Provides the geologic timescale color scheme.

Usage

```
scale_colour_stratigraphy(..., lang = "en", aesthetics = "colour")
```

```
scale_color_stratigraphy(..., lang = "en", aesthetics = "colour")
```

```
scale_fill_stratigraphy(..., lang = "en", aesthetics = "fill")
```

Arguments

...	Arguments passed on to <code>ggplot2::discrete_scale()</code> .
lang	A character string specifying the language for the color names (see details). It must be one of "en" (english, the default) or "fr" (french).
aesthetics	A character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

Values will be matched based on the unit names.

Value

A [discrete](#) scale.

Author(s)

N. Frerebeau

References

[Commission for the Geological Map of the World.](#)

See Also

Other themed colour schemes: [scale_colour_land\(\)](#), [scale_colour_soil\(\)](#)

Other qualitative colour schemes: [scale_colour_land\(\)](#), [scale_colour_soil\(\)](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#)

Examples

```

library(ggplot2)

strati <- data.frame(
  name = c("Phanerozoic", "Paleozoic", "Cambrian", "Ordovician", "Silurian",
           "Devonian", "Carboniferous", "Mesozoic", "Triassic", "Cretaceous",
           "Jurassic", "Cenozoic", "Paleogene", "Neogene", "Quaternary"),
  type = c("Eon", "Era", "Period", "Period", "Period", "Period", "Period",
           "Era", "Period", "Period", "Period", "Era", "Period", "Period",
           "Period"),
  start = c(541, 541, 541, 485, 444, 419, 359,
            252, 252, 201, 145, 66, 66, 23, 2.6),
  end = c(0, 252, 485, 444, 419, 359, 252,
          66, 201, 145, 66, 2.6, 23, 2.6, 0)
)

ggplot2::ggplot(strati, ggplot2::aes(fill = name)) +
  ggplot2::geom_rect(aes(xmin = rep(0, 15), xmax = rep(1, 15),
                        ymin = start, ymax = end)) +
  ggplot2::scale_y_reverse() +
  ggplot2::facet_grid(. ~ type) +
  scale_fill_stratigraphy(name = "Stratigraphy")

```

scale_crameri_cyclic *Fabio Crameri's Cyclic Color Schemes for ggplot2*

Description

Provides cyclic color scales from Fabio Crameri's *Scientific colour*.

Usage

```

scale_colour_broc0(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

scale_color_broc0(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

```

```
scale_fill_broc0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)  
  
scale_colour_cork0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_color_cork0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_fill_cork0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)  
  
scale_colour_vik0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_color_vik0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_vik0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)  
  
scale_colour_roma0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_color_roma0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_fill_roma0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)  
  
scale_colour_bam0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_color_bam0(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_bam0(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "fill"
)
```

Arguments

...	Arguments passed to <code>ggplot2::continuous_scale()</code> .
reverse	A logical scalar. Should the resulting vector of colors be reversed?
range	A length-two numeric vector specifying the fraction of the scheme's color domain to keep.
discrete	A logical scalar: should the color scheme be used as a discrete scale?
aesthetics	A character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

If more colors than defined are needed from a given scheme, the color coordinates are linearly interpolated to provide a continuous version of the scheme.

Note that the default color for NA can be overridden by passing a value to `ggplot2::continuous_scale()`.

Available schemes:

- broc0
- cork0
- vik0
- roma0
- bam0

Value

A **continuous** scale.

Author(s)

N. Frerebeau

Source

Crameri, F. (2021). Scientific colour maps. *Zenodo*, v7.0. doi: [10.5281/zenodo.4491293](https://doi.org/10.5281/zenodo.4491293)

References

- Crameri, F. (2018). Geodynamic diagnostics, scientific visualisation and StagLab 3.0. *Geosci. Model Dev.*, 11, 2541-2562. doi: [10.5194/gmd1125412018](https://doi.org/10.5194/gmd1125412018)
- Crameri, F., Shephard, G. E. & Heron, P. J. (2020). The misuse of colour in science communication. *Nature Communications*, 11, 5444. doi: [10.1038/s41467020191607](https://doi.org/10.1038/s41467020191607)

See Also

Other colour-blind safe colour schemes: [scale_crameri_diverging](#), [scale_crameri_mutlisequential](#), [scale_crameri_sequential](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#), [scale_tol_diverging](#), [scale_tol_sequential](#)

Other Fabio Crameri's colour schemes: [scale_crameri_diverging](#), [scale_crameri_mutlisequential](#), [scale_crameri_sequential](#)

Examples

```
data(economics, package = "ggplot2")

ggplot2::ggplot(economics, ggplot2::aes(psavert, pce, colour = unemploy)) +
  ggplot2::geom_point() +
  scale_colour_broc(reverse = TRUE, midpoint = 12000)

ggplot2::ggplot(economics, ggplot2::aes(psavert, pce, colour = unemploy)) +
  ggplot2::geom_point() +
  scale_colour_berlin(midpoint = 9000)
```

scale_crameri_diverging

Fabio Crameri's Diverging Colour Schemes for ggplot2

Description

Provides diverging colour scales from Fabio Crameri's *Scientific colour*.

Usage

```
scale_colour_broc(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  discrete = FALSE,
  aesthetics = "colour"
)

scale_color_broc(
  ...,
```

```
reverse = FALSE,  
range = c(0, 1),  
midpoint = 0,  
discrete = FALSE,  
aesthetics = "colour"  
)
```

```
scale_fill_broc(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_cork(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_cork(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_cork(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_vik(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,
```

```
    discrete = FALSE,  
    aesthetics = "colour"  
  )
```

```
scale_color_vik(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_vik(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_lisbon(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_lisbon(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_lisbon(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_tofino(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_tofino(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_tofino(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_berlin(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_berlin(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_berlin(  
  ...,
```

```
reverse = FALSE,  
range = c(0, 1),  
midpoint = 0,  
discrete = FALSE,  
aesthetics = "fill"  
)  
  
scale_colour_roma(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_color_roma(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_fill_roma(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "fill"  
)  
  
scale_colour_bam(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)  
  
scale_color_bam(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,
```

```

    discrete = FALSE,
    aesthetics = "colour"
  )

scale_fill_bam(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  discrete = FALSE,
  aesthetics = "fill"
)

scale_colour_vanimo(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  discrete = FALSE,
  aesthetics = "colour"
)

scale_color_vanimo(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  discrete = FALSE,
  aesthetics = "colour"
)

scale_fill_vanimo(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  discrete = FALSE,
  aesthetics = "fill"
)

```

Arguments

...	Arguments passed to <code>ggplot2::continuous_scale()</code> .
reverse	A logical scalar. Should the resulting vector of colors be reversed?
range	A length-two numeric vector specifying the fraction of the scheme's color domain to keep.
midpoint	A length-one numeric vector giving the midpoint (in data value) of the diverging scale. Defaults to 0.

discrete A [logical](#) scalar: should the color scheme be used as a discrete scale?
aesthetics A [character](#) string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

If more colors than defined are needed from a given scheme, the color coordinates are linearly interpolated to provide a continuous version of the scheme. Note that the default color for NA can be overridden by passing a value to `ggplot2::continuous_scale()`.

Available schemes:

- broc
- cork
- vik
- lisbon
- tofino
- berlin
- roma
- bam
- vanimo

Value

A [continuous](#) scale.

Author(s)

N. Frerebeau

Source

Crameri, F. (2021). Scientific colour maps. *Zenodo*, v7.0. doi: [10.5281/zenodo.4491293](https://doi.org/10.5281/zenodo.4491293)

References

Crameri, F. (2018). Geodynamic diagnostics, scientific visualisation and StagLab 3.0. *Geosci. Model Dev.*, 11, 2541-2562. doi: [10.5194/gmd1125412018](https://doi.org/10.5194/gmd1125412018)

Crameri, F., Shephard, G. E. & Heron, P. J. (2020). The misuse of colour in science communication. *Nature Communications*, 11, 5444. doi: [10.1038/s41467020191607](https://doi.org/10.1038/s41467020191607)

See Also

Other colour-blind safe colour schemes: [scale_crameri_cyclic](#), [scale_crameri_mutlisequential](#), [scale_crameri_sequential](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#), [scale_tol_diverging](#), [scale_tol_sequential](#)

Other diverging colour schemes: [scale_tol_diverging](#)

Other Fabio Crameri's colour schemes: [scale_crameri_cyclic](#), [scale_crameri_mutlisequential](#), [scale_crameri_sequential](#)

Examples

```

data(economics, package = "ggplot2")

ggplot2::ggplot(economics, ggplot2::aes(psavert, pce, colour = unemploy)) +
  ggplot2::geom_point() +
  scale_colour_broc(reverse = TRUE, midpoint = 12000)

ggplot2::ggplot(economics, ggplot2::aes(psavert, pce, colour = unemploy)) +
  ggplot2::geom_point() +
  scale_colour_berlin(midpoint = 9000)

```

scale_crameri_mutlisequential

*Fabio Crameri's Multi-Sequential Color Schemes for **ggplot2***

Description

Provides multi-sequential color scales from Fabio Crameri's *Scientific colour*.

Usage

```

scale_colour_oleron(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  aesthetics = "colour"
)

scale_color_oleron(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  aesthetics = "colour"
)

scale_fill_oleron(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  aesthetics = "fill"
)

scale_colour_bukavu(
  ...,

```



```
reverse = FALSE,  
range = c(0, 1),  
midpoint = 0,  
aesthetics = "colour"  
)  
  
scale_color_bukavu(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  aesthetics = "colour"  
)  
  
scale_fill_bukavu(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  aesthetics = "fill"  
)  
  
scale_colour_fes(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  aesthetics = "colour"  
)  
  
scale_color_fes(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  aesthetics = "colour"  
)  
  
scale_fill_fes(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  aesthetics = "fill"  
)
```

Arguments

... Arguments passed to `ggplot2::continuous_scale()`.

reverse	A logical scalar. Should the resulting vector of colors be reversed?
range	A length-two numeric vector specifying the fraction of the scheme's color domain to keep.
midpoint	A length-one numeric vector giving the midpoint (in data value) of the diverging scale. Defaults to 0.
aesthetics	A character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

If more colors than defined are needed from a given scheme, the color coordinates are linearly interpolated to provide a continuous version of the scheme.

Note that the default color for NA can be overridden by passing a value to `ggplot2::continuous_scale()`.

Available schemes:

- oleron
- bukavu
- fes

Value

A [continuous](#) scale.

Author(s)

N. Frerebeau

Source

Crameri, F. (2021). Scientific colour maps. *Zenodo*, v7.0. doi: [10.5281/zenodo.4491293](https://doi.org/10.5281/zenodo.4491293)

References

Crameri, F. (2018). Geodynamic diagnostics, scientific visualisation and StagLab 3.0. *Geosci. Model Dev.*, 11, 2541-2562. doi: [10.5194/gmd1125412018](https://doi.org/10.5194/gmd1125412018)

Crameri, F., Shephard, G. E. & Heron, P. J. (2020). The misuse of colour in science communication. *Nature Communications*, 11, 5444. doi: [10.1038/s41467020191607](https://doi.org/10.1038/s41467020191607)

See Also

Other colour-blind safe colour schemes: [scale_crameri_cyclic](#), [scale_crameri_diverging](#), [scale_crameri_sequential](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#), [scale_tol_diverging](#), [scale_tol_sequential](#)

Other Fabio Crameri's colour schemes: [scale_crameri_cyclic](#), [scale_crameri_diverging](#), [scale_crameri_sequential](#)

Examples

```
data(volcano)

volcan <- data.frame(
  x = rep(1:ncol(volcano), each = nrow(volcano)),
  y = rep(1:nrow(volcano), times = ncol(volcano)),
  z = as.numeric(volcano)
)

ggplot2::ggplot(volcan, ggplot2::aes(x, y, fill = z)) +
  ggplot2::geom_raster() +
  scale_fill_oleron(midpoint = 125)
```

scale_crameri_sequential

Fabio Crameri's Sequential Color Schemes for ggplot2

Description

Provides sequential color scales from Fabio Crameri's *Scientific colour*.

Usage

```
scale_colour_batlow(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

scale_color_batlow(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

scale_fill_batlow(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "fill"
)
```

```
scale_colour_batlowW(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_batlowW(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_batlowW(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_batlowK(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_batlowK(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_batlowK(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_devon(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_devon(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_devon(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_lajolla(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_lajolla(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_lajolla(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_bamako(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_bamako(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_bamako(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_davos(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_davos(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_davos(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_bilbao(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_bilbao(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_bilbao(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_nuuk(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_nuuk(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_nuuk(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_oslo(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_oslo(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_oslo(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_grayC(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_grayC(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_grayC(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```



```
scale_colour_hawaii(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_hawaii(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_hawaii(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_lapaz(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_lapaz(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_lapaz(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_tokyo(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_tokyo(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_tokyo(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_buda(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_buda(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_buda(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_acton(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_acton(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_acton(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_turku(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_turku(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_turku(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```

scale_colour_imola(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

scale_color_imola(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

scale_fill_imola(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "fill"
)

```

Arguments

...	Arguments passed to <code>ggplot2::continuous_scale()</code> .
reverse	A logical scalar. Should the resulting vector of colors be reversed?
range	A length-two numeric vector specifying the fraction of the scheme's color domain to keep.
discrete	A logical scalar: should the color scheme be used as a discrete scale?
aesthetics	A character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

If more colors than defined are needed from a given scheme, the color coordinates are linearly interpolated to provide a continuous version of the scheme.

Note that the default color for NA can be overridden by passing a value to `ggplot2::continuous_scale()`.

Available schemes:

- batlow
- batlowW
- batlowK
- devon
- lajolla

- bamako
- davos
- bilbao
- nuuk
- oslo
- grayC
- hawaii
- lapaz
- tokyo
- buda
- acton
- turku
- imola

Value

A [continuous](#) scale.

Author(s)

N. Frerebeau

Source

Crameri, F. (2021). Scientific colour maps. *Zenodo*, v7.0. doi: [10.5281/zenodo.4491293](https://doi.org/10.5281/zenodo.4491293)

References

Crameri, F. (2018). Geodynamic diagnostics, scientific visualisation and StagLab 3.0. *Geosci. Model Dev.*, 11, 2541-2562. doi: [10.5194/gmd1125412018](https://doi.org/10.5194/gmd1125412018)

Crameri, F., Shephard, G. E. & Heron, P. J. (2020). The misuse of colour in science communication. *Nature Communications*, 11, 5444. doi: [10.1038/s41467020191607](https://doi.org/10.1038/s41467020191607)

See Also

Other colour-blind safe colour schemes: [scale_crameri_cyclic](#), [scale_crameri_diverging](#), [scale_crameri_mutlisequential](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#), [scale_tol_diverging](#), [scale_tol_sequential](#)

Other sequential colour schemes: [scale_tol_sequential](#)

Other Fabio Crameri's colour schemes: [scale_crameri_cyclic](#), [scale_crameri_diverging](#), [scale_crameri_mutlisequential](#)

Examples

```
data(faithfuld, package = "ggplot2")

ggplot2::ggplot(faithfuld, ggplot2::aes(waiting, eruptions, fill = density)) +
  ggplot2::geom_raster() +
  scale_fill_batlow()

ggplot2::ggplot(faithfuld, ggplot2::aes(waiting, eruptions, fill = density)) +
  ggplot2::geom_raster() +
  scale_fill_bamako()

ggplot2::ggplot(faithfuld, ggplot2::aes(waiting, eruptions, fill = density)) +
  ggplot2::geom_raster() +
  scale_fill_hawaii(reverse = TRUE)
```

scale_okabeito_discrete

Okabe and Ito's Discrete Color Scheme for ggplot2

Description

Provides the qualitative color scale from Okabe and Ito 2008.

Usage

```
scale_colour_okabeito(..., reverse = FALSE, aesthetics = "colour")

scale_color_okabeito(..., reverse = FALSE, aesthetics = "colour")

scale_fill_okabeito(..., reverse = FALSE, aesthetics = "fill")
```

Arguments

...	Arguments passed to <code>ggplot2::discrete_scale()</code> .
reverse	A logical scalar. Should the resulting vector of colors be reversed?
aesthetics	A character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

This qualitative color scheme is used as given (no interpolation): colors are picked up to the maximum number of supported values (8).

Value

A [discrete](#) scale.

Author(s)

N. Frerebeau

References

Okabe, M. & Ito, K. (2008). *Color Universal Design (CUD): How to Make Figures and Presentations That Are Friendly to Colorblind People*. URL: <https://jfly.uni-koeln.de/color/>.

See Also

Other colour-blind safe colour schemes: [scale_crameri_cyclic](#), [scale_crameri_diverging](#), [scale_crameri_mutlisequential](#), [scale_crameri_sequential](#), [scale_tol_discrete](#), [scale_tol_diverging](#), [scale_tol_sequential](#)

Other qualitative colour schemes: [scale_colour_land\(\)](#), [scale_colour_soil\(\)](#), [scale_colour_stratigraphy\(\)](#), [scale_tol_discrete](#)

Examples

```
library(ggplot2)

ggplot2::ggplot(mpg, ggplot2::aes(displ, hwy, colour = class)) +
  ggplot2::geom_point() +
  scale_colour_okabeito()
```

scale_tol_discrete *Paul Tol's Discrete Color Schemes for ggplot2*

Description

Provides qualitative color scales from Paul Tol's *Colour Schemes*.

Usage

```
scale_colour_bright(..., reverse = FALSE, aesthetics = "colour")
scale_color_bright(..., reverse = FALSE, aesthetics = "colour")
scale_fill_bright(..., reverse = FALSE, aesthetics = "fill")
scale_colour_contrast(..., reverse = FALSE, aesthetics = "colour")
scale_color_contrast(..., reverse = FALSE, aesthetics = "colour")
scale_fill_contrast(..., reverse = FALSE, aesthetics = "fill")
scale_colour_vibrant(..., reverse = FALSE, aesthetics = "colour")
```

```

scale_color_vibrant(..., reverse = FALSE, aesthetics = "colour")
scale_fill_vibrant(..., reverse = FALSE, aesthetics = "fill")
scale_colour_muted(..., reverse = FALSE, aesthetics = "colour")
scale_color_muted(..., reverse = FALSE, aesthetics = "colour")
scale_fill_muted(..., reverse = FALSE, aesthetics = "fill")
scale_colour_pale(..., reverse = FALSE, aesthetics = "colour")
scale_color_pale(..., reverse = FALSE, aesthetics = "colour")
scale_fill_pale(..., reverse = FALSE, aesthetics = "fill")
scale_colour_dark(..., reverse = FALSE, aesthetics = "colour")
scale_color_dark(..., reverse = FALSE, aesthetics = "colour")
scale_fill_dark(..., reverse = FALSE, aesthetics = "fill")
scale_colour_light(..., reverse = FALSE, aesthetics = "colour")
scale_color_light(..., reverse = FALSE, aesthetics = "colour")
scale_fill_light(..., reverse = FALSE, aesthetics = "fill")

```

Arguments

... Arguments passed to `ggplot2::discrete_scale()`.

reverse A **logical** scalar. Should the resulting vector of colors be reversed?

aesthetics A **character** string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

The qualitative color schemes are used as given (no interpolation): colors are picked up to the maximum number of supported values.

Palette	Max.
bright	7
contrast	3
vibrant	7
muted	9
pale	6
dark	6
light	9

Value

A [discrete](#) scale.

Qualitative color schemes

According to Paul Tol's technical note, the bright, contrast, vibrant and muted color schemes are colorblind safe.

The light color scheme is reasonably distinct for both normal or colorblind vision and is intended to fill labeled cells.

The pale and dark schemes are not very distinct in either normal or colorblind vision and should be used as a text background or to highlight a cell in a table.

Refer to the original document for details about the recommended uses (see references).

Author(s)

N. Frerebeau

References

Tol, P. (2018). *Colour Schemes*. SRON. Technical Note No. SRON/EPS/TN/09-002, issue 3.1. URL: <https://personal.sron.nl/~pault/data/colourschemes.pdf>

See Also

Other colour-blind safe colour schemes: [scale_crameri_cyclic](#), [scale_crameri_diverging](#), [scale_crameri_mutlisequential](#), [scale_crameri_sequential](#), [scale_okabeito_discrete](#), [scale_tol_diverging](#), [scale_tol_sequential](#)

Other qualitative colour schemes: [scale_colour_land\(\)](#), [scale_colour_soil\(\)](#), [scale_colour_stratigraphy\(\)](#), [scale_okabeito_discrete](#)

Other Paul Tol's colour schemes: [scale_tol_diverging](#), [scale_tol_sequential](#)

Examples

```
library(ggplot2)

ggplot2::ggplot(mpg, ggplot2::aes(displ, hwy, colour = class)) +
  ggplot2::geom_point() +
  scale_colour_bright()

ggplot2::ggplot(mpg, ggplot2::aes(displ, hwy, colour = class)) +
  ggplot2::geom_point() +
  scale_colour_vibrant()

ggplot2::ggplot(diamonds, ggplot2::aes(clarity, fill = cut)) +
  ggplot2::geom_bar() +
  scale_fill_muted()

ggplot2::ggplot(diamonds, ggplot2::aes(clarity, fill = cut)) +
  ggplot2::geom_bar() +
```

```
scale_fill_pale()

ggplot2::ggplot(diamonds, ggplot2::aes(clarity, fill = cut)) +
  ggplot2::geom_bar() +
  scale_fill_dark()

ggplot2::ggplot(diamonds, ggplot2::aes(clarity, fill = cut)) +
  ggplot2::geom_bar() +
  scale_fill_light()
```

scale_tol_diverging *Paul Tol's Diverging Colour Schemes for ggplot2*

Description

Provides diverging colour scales from Paul Tol's *Colour Schemes*.

Usage

```
scale_colour_sunset(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  discrete = FALSE,
  aesthetics = "colour"
)

scale_color_sunset(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  discrete = FALSE,
  aesthetics = "colour"
)

scale_fill_sunset(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  midpoint = 0,
  discrete = FALSE,
  aesthetics = "fill"
)

scale_colour_BuRd(
  ...,
```

```
reverse = FALSE,  
range = c(0, 1),  
midpoint = 0,  
discrete = FALSE,  
aesthetics = "colour"  
)
```

```
scale_color_BuRd(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_BuRd(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "fill"  
)
```

```
scale_colour_PRGn(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_color_PRGn(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,  
  discrete = FALSE,  
  aesthetics = "colour"  
)
```

```
scale_fill_PRGn(  
  ...,  
  reverse = FALSE,  
  range = c(0, 1),  
  midpoint = 0,
```

```

discrete = FALSE,
aesthetics = "fill"
)

```

Arguments

...	Arguments passed to <code>ggplot2::continuous_scale()</code> .
reverse	A logical scalar. Should the resulting vector of colors be reversed?
range	A length-two numeric vector specifying the fraction of the scheme's color domain to keep.
midpoint	A length-one numeric vector giving the midpoint (in data value) of the diverging scale. Defaults to 0.
discrete	A logical scalar: should the color scheme be used as a discrete scale? If TRUE, it is a departure from Paul Tol's recommendations and likely a very poor use of color.
aesthetics	A character string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

If more colors than defined are needed from a given scheme, the color coordinates are linearly interpolated to provide a continuous version of the scheme. Note that the default color for NA can be overridden by passing a value to `ggplot2::continuous_scale()`.

Palette	Max. colours	NA value
sunset	11	#FFFFFF
BuRd	9	#FFEE99
PRGn	9	#FFEE99

Value

A **continuous** scale.

Author(s)

N. Frerebeau

References

Tol, P. (2018). *Colour Schemes*. SRON. Technical Note No. SRON/EPS/TN/09-002, issue 3.1. URL: <https://personal.sron.nl/~pault/data/colourschemes.pdf>

See Also

Other colour-blind safe colour schemes: [scale_crameri_cyclic](#), [scale_crameri_diverging](#), [scale_crameri_mutlisequential](#), [scale_crameri_sequential](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#), [scale_tol_sequential](#)

Other diverging colour schemes: [scale_crameri_diverging](#)

Other Paul Tol's colour schemes: [scale_tol_discrete](#), [scale_tol_sequential](#)

Examples

```
data(economics, package = "ggplot2")

ggplot2::ggplot(economics, ggplot2::aes(psavert, pce, colour = unemploy)) +
  ggplot2::geom_point() +
  scale_color_sunset(reverse = TRUE, midpoint = 12000)

ggplot2::ggplot(economics, ggplot2::aes(psavert, pce, colour = unemploy)) +
  ggplot2::geom_point() +
  scale_color_BuRd(midpoint = 9000)

ggplot2::ggplot(economics, ggplot2::aes(psavert, pce, colour = unemploy)) +
  ggplot2::geom_point() +
  scale_color_PRGn(midpoint = 9000, range = c(0.25, 1))
```

scale_tol_sequential *Paul Tol's Sequential Color Schemes for ggplot2*

Description

Provides sequential color scales from Paul Tol's *Colour Schemes*.

Usage

```
scale_colour_YlOrBr(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)
```

```
scale_color_YlOrBr(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)
```

```
scale_fill_YlOrBr(
  ...,
  reverse = FALSE,
  range = c(0, 1),
)
```

```
    discrete = FALSE,
    aesthetics = "fill"
  )

scale_colour_iridescent(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

scale_color_iridescent(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

scale_fill_iridescent(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "fill"
)

scale_colour_discreterainbow(..., reverse = FALSE, aesthetics = "colour")
scale_color_discreterainbow(..., reverse = FALSE, aesthetics = "colour")
scale_fill_discreterainbow(..., reverse = FALSE, aesthetics = "fill")

scale_colour_smoothrainbow(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)

scale_color_smoothrainbow(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "colour"
)
```

```

)

scale_fill_smoothrainbow(
  ...,
  reverse = FALSE,
  range = c(0, 1),
  discrete = FALSE,
  aesthetics = "fill"
)

```

Arguments

... Arguments passed to `ggplot2::continuous_scale()`.

reverse A **logical** scalar. Should the resulting vector of colors be reversed?

range A length-two **numeric** vector specifying the fraction of the scheme's color domain to keep.

discrete A **logical** scalar: should the color scheme be used as a discrete scale? If TRUE, it is a departure from Paul Tol's recommendations and likely a very poor use of color.

aesthetics A **character** string or vector of character strings listing the name(s) of the aesthetic(s) that this scale works with.

Details

If more colors than defined are needed from a given scheme, the color coordinates are linearly interpolated to provide a continuous version of the scheme, with the exception of the discrete rainbow scheme (see below).

Note that the default color for NA can be overridden by passing a value to `ggplot2::continuous_scale()`.

Palette	Max. colors	NA value
YlOrBr	9	#888888
iridescent	23	#999999
discrete rainbow	23	#777777
smooth rainbow	34	#666666

Value

A **continuous** scale.

Rainbow color scheme

As a general rule, ordered data should not be represented using a rainbow scheme. There are three main arguments against such use (Tol 2018):

- The spectral order of visible light carries no inherent magnitude message.
- Some bands of almost constant hue with sharp transitions between them, can be perceived as jumps in the data.

- Colour-blind people have difficulty distinguishing some colours of the rainbow.

If such use cannot be avoided, Paul Tol's technical note provides two colour schemes that are reasonably clear in colour-blind vision. To remain colour-blind safe, these two schemes must comply with the following conditions:

discrete rainbow This scheme must not be interpolated.

smooth rainbow This scheme does not have to be used over the full range.

Author(s)

N. Frerebeau

References

Tol, P. (2018). *Colour Schemes*. SRON. Technical Note No. SRON/EPS/TN/09-002, issue 3.1. URL: <https://personal.sron.nl/~pault/data/colourschemes.pdf>

See Also

Other colour-blind safe colour schemes: [scale_crameri_cyclic](#), [scale_crameri_diverging](#), [scale_crameri_mutlisequential](#), [scale_crameri_sequential](#), [scale_okabeito_discrete](#), [scale_tol_discrete](#), [scale_tol_diverging](#)

Other sequential colour schemes: [scale_crameri_sequential](#)

Other Paul Tol's colour schemes: [scale_tol_discrete](#), [scale_tol_diverging](#)

Examples

```
data(faithfuld, package = "ggplot2")
```

```
ggplot2::ggplot(faithfuld, ggplot2::aes(waiting, eruptions, fill = density)) +
  ggplot2::geom_raster() +
  scale_fill_YlOrBr()
```

```
ggplot2::ggplot(faithfuld, ggplot2::aes(waiting, eruptions, fill = density)) +
  ggplot2::geom_raster() +
  scale_fill_iridescent(reverse = TRUE)
```

```
ggplot2::ggplot(faithfuld, ggplot2::aes(waiting, eruptions, fill = density)) +
  ggplot2::geom_raster() +
  scale_fill_smoothrainbow(range = c(0.25, 1))
```


Index

- * **Fabio Crameri's colour schemes**
 - scale_crameri_cyclic, 13
 - scale_crameri_diverging, 17
 - scale_crameri_mutlisequential, 24
 - scale_crameri_sequential, 27
- * **Okabe and Ito's colour scheme**
 - scale_okabeito_discrete, 38
- * **Paul Tol's colour schemes**
 - scale_tol_discrete, 39
 - scale_tol_diverging, 42
 - scale_tol_sequential, 45
- * **color**
 - colour, 2
- * **colour palettes**
 - colour, 2
- * **colour-blind safe colour schemes**
 - scale_crameri_cyclic, 13
 - scale_crameri_diverging, 17
 - scale_crameri_mutlisequential, 24
 - scale_crameri_sequential, 27
 - scale_okabeito_discrete, 38
 - scale_tol_discrete, 39
 - scale_tol_diverging, 42
 - scale_tol_sequential, 45
- * **cyclic colour schemes**
 - scale_crameri_cyclic, 13
- * **diagnostic tools**
 - compare, 5
 - convert, 6
 - plot, 8
- * **diverging colour schemes**
 - scale_crameri_diverging, 17
 - scale_tol_diverging, 42
- * **multi sequential colour schemes**
 - scale_crameri_mutlisequential, 24
- * **qualitative colour schemes**
 - scale_colour_land, 9
 - scale_colour_soil, 10
 - scale_colour_stratigraphy, 12
 - scale_okabeito_discrete, 38
 - scale_tol_discrete, 39
- * **sequential colour schemes**
 - scale_crameri_sequential, 27
 - scale_tol_sequential, 45
- * **themed colour schemes**
 - scale_colour_land, 9
 - scale_colour_soil, 10
 - scale_colour_stratigraphy, 12
- character, 2, 3, 5, 6, 8, 9, 11, 12, 16, 23, 26, 36, 38, 40, 44, 47
- color (colour), 2
- colorRampPalette, 2
- colour, 2
- colour(), 6, 8
- compare, 5, 7, 8
- continuous, 16, 23, 26, 37, 44, 47
- convert, 6, 6, 8
- discrete, 9, 11, 12, 38, 41
- distance matrix, 5
- function, 6, 7
- ggplot2::continuous_scale(), 16, 22, 23, 25, 26, 36, 44, 47
- ggplot2::discrete_scale(), 9, 11, 12, 38, 40
- graphics::par(), 8
- integer, 3, 5
- logical, 2, 3, 5, 8, 16, 22, 23, 26, 36, 38, 40, 44, 47
- numeric, 8, 16, 22, 26, 36, 44, 47
- plot, 6, 7, 8
- plot_map(plot), 8
- plot_scheme(plot), 8

- plot_scheme_colorblind(plot), 8
- plot_scheme_colourblind(plot), 8
- scale_color_acton
(scale_crameri_sequential), 27
- scale_color_bam
(scale_crameri_diverging), 17
- scale_color_bamako
(scale_crameri_sequential), 27
- scale_color_bam0
(scale_crameri_cyclic), 13
- scale_color_batlow
(scale_crameri_sequential), 27
- scale_color_batlowK
(scale_crameri_sequential), 27
- scale_color_batlowW
(scale_crameri_sequential), 27
- scale_color_berlin
(scale_crameri_diverging), 17
- scale_color_bilbao
(scale_crameri_sequential), 27
- scale_color_bright
(scale_tol_discrete), 39
- scale_color_broc
(scale_crameri_diverging), 17
- scale_color_broc0
(scale_crameri_cyclic), 13
- scale_color_buda
(scale_crameri_sequential), 27
- scale_color_bukavu
(scale_crameri_mutlisequential),
24
- scale_color_BuRd (scale_tol_diverging),
42
- scale_color_contrast
(scale_tol_discrete), 39
- scale_color_cork
(scale_crameri_diverging), 17
- scale_color_cork0
(scale_crameri_cyclic), 13
- scale_color_dark (scale_tol_discrete),
39
- scale_color_davos
(scale_crameri_sequential), 27
- scale_color_devon
(scale_crameri_sequential), 27
- scale_color_discreterainbow
(scale_tol_sequential), 45
- scale_color_fes
(scale_crameri_mutlisequential),
24
- scale_color_grayC
(scale_crameri_sequential), 27
- scale_color_hawaii
(scale_crameri_sequential), 27
- scale_color_imola
(scale_crameri_sequential), 27
- scale_color_iridescent
(scale_tol_sequential), 45
- scale_color_lajolla
(scale_crameri_sequential), 27
- scale_color_land (scale_colour_land), 9
- scale_color_lapaz
(scale_crameri_sequential), 27
- scale_color_light (scale_tol_discrete),
39
- scale_color_lisbon
(scale_crameri_diverging), 17
- scale_color_muted (scale_tol_discrete),
39
- scale_color_nuuk
(scale_crameri_sequential), 27
- scale_color_okabeito
(scale_okabeito_discrete), 38
- scale_color_oleron
(scale_crameri_mutlisequential),
24
- scale_color_oslo
(scale_crameri_sequential), 27
- scale_color_pale (scale_tol_discrete),
39
- scale_color_PRGn (scale_tol_diverging),
42
- scale_color_roma
(scale_crameri_diverging), 17
- scale_color_roma0
(scale_crameri_cyclic), 13
- scale_color_smoothrainbow
(scale_tol_sequential), 45
- scale_color_soil (scale_colour_soil), 10
- scale_color_stratigraphy
(scale_colour_stratigraphy), 12
- scale_color_sunset
(scale_tol_diverging), 42
- scale_color_tofino
(scale_crameri_diverging), 17

- scale_color_tokyo
(scale_crameri_sequential), 27
- scale_color_turku
(scale_crameri_sequential), 27
- scale_color_vanimo
(scale_crameri_diverging), 17
- scale_color_vibrant
(scale_tol_discrete), 39
- scale_color_vik
(scale_crameri_diverging), 17
- scale_color_vik0
(scale_crameri_cyclic), 13
- scale_color_YlOrBr
(scale_tol_sequential), 45
- scale_colour_acton
(scale_crameri_sequential), 27
- scale_colour_bam
(scale_crameri_diverging), 17
- scale_colour_bamako
(scale_crameri_sequential), 27
- scale_colour_bam0
(scale_crameri_cyclic), 13
- scale_colour_batlow
(scale_crameri_sequential), 27
- scale_colour_batlowK
(scale_crameri_sequential), 27
- scale_colour_batlowW
(scale_crameri_sequential), 27
- scale_colour_berlin
(scale_crameri_diverging), 17
- scale_colour_bilbao
(scale_crameri_sequential), 27
- scale_colour_bright
(scale_tol_discrete), 39
- scale_colour_broc
(scale_crameri_diverging), 17
- scale_colour_broc0
(scale_crameri_cyclic), 13
- scale_colour_buda
(scale_crameri_sequential), 27
- scale_colour_bukavu
(scale_crameri_mutlisequential),
24
- scale_colour_BuRd
(scale_tol_diverging), 42
- scale_colour_contrast
(scale_tol_discrete), 39
- scale_colour_cork
(scale_crameri_diverging), 17
- scale_colour_cork0
(scale_crameri_cyclic), 13
- scale_colour_dark (scale_tol_discrete),
39
- scale_colour_davos
(scale_crameri_sequential), 27
- scale_colour_devon
(scale_crameri_sequential), 27
- scale_colour_discreterainbow
(scale_tol_sequential), 45
- scale_colour_fes
(scale_crameri_mutlisequential),
24
- scale_colour_grayC
(scale_crameri_sequential), 27
- scale_colour_hawaii
(scale_crameri_sequential), 27
- scale_colour_imola
(scale_crameri_sequential), 27
- scale_colour_iridescent
(scale_tol_sequential), 45
- scale_colour_lajolla
(scale_crameri_sequential), 27
- scale_colour_land, 9, 11, 12, 39, 41
- scale_colour_lapaz
(scale_crameri_sequential), 27
- scale_colour_light
(scale_tol_discrete), 39
- scale_colour_lisbon
(scale_crameri_diverging), 17
- scale_colour_muted
(scale_tol_discrete), 39
- scale_colour_nuuk
(scale_crameri_sequential), 27
- scale_colour_okabeito
(scale_okabeito_discrete), 38
- scale_colour_oleron
(scale_crameri_mutlisequential),
24
- scale_colour_oslo
(scale_crameri_sequential), 27
- scale_colour_pale (scale_tol_discrete),
39
- scale_colour_PRGn
(scale_tol_diverging), 42
- scale_colour_roma
(scale_crameri_diverging), 17

- scale_colour_roma0
(scale_crameri_cyclic), 13
- scale_colour_smoothrainbow
(scale_tol_sequential), 45
- scale_colour_soil, 10, 10, 12, 39, 41
- scale_colour_stratigraphy, 10, 11, 12, 39, 41
- scale_colour_sunset
(scale_tol_diverging), 42
- scale_colour_tofino
(scale_crameri_diverging), 17
- scale_colour_tokyo
(scale_crameri_sequential), 27
- scale_colour_turku
(scale_crameri_sequential), 27
- scale_colour_vanimo
(scale_crameri_diverging), 17
- scale_colour_vibrant
(scale_tol_discrete), 39
- scale_colour_vik
(scale_crameri_diverging), 17
- scale_colour_vik0
(scale_crameri_cyclic), 13
- scale_colour_YlOrBr
(scale_tol_sequential), 45
- scale_crameri_cyclic, 13, 23, 26, 37, 39, 41, 44, 48
- scale_crameri_diverging, 17, 17, 26, 37, 39, 41, 44, 45, 48
- scale_crameri_mutlisequential, 17, 23, 24, 37, 39, 41, 44, 48
- scale_crameri_sequential, 17, 23, 26, 27, 39, 41, 44, 48
- scale_fill_acton
(scale_crameri_sequential), 27
- scale_fill_bam
(scale_crameri_diverging), 17
- scale_fill_bamako
(scale_crameri_sequential), 27
- scale_fill_bam0 (scale_crameri_cyclic), 13
- scale_fill_batlow
(scale_crameri_sequential), 27
- scale_fill_batlowK
(scale_crameri_sequential), 27
- scale_fill_batlowW
(scale_crameri_sequential), 27
- scale_fill_berlin
(scale_crameri_diverging), 17
- scale_fill_bilbao
(scale_crameri_sequential), 27
- scale_fill_bright (scale_tol_discrete), 39
- scale_fill_broc
(scale_crameri_diverging), 17
- scale_fill_broc0
(scale_crameri_cyclic), 13
- scale_fill_buda
(scale_crameri_sequential), 27
- scale_fill_bukavu
(scale_crameri_mutlisequential), 24
- scale_fill_BuRd (scale_tol_diverging), 42
- scale_fill_contrast
(scale_tol_discrete), 39
- scale_fill_cork
(scale_crameri_diverging), 17
- scale_fill_cork0
(scale_crameri_cyclic), 13
- scale_fill_dark (scale_tol_discrete), 39
- scale_fill_davos
(scale_crameri_sequential), 27
- scale_fill_devon
(scale_crameri_sequential), 27
- scale_fill_discreterainbow
(scale_tol_sequential), 45
- scale_fill_fes
(scale_crameri_mutlisequential), 24
- scale_fill_grayC
(scale_crameri_sequential), 27
- scale_fill_hawaii
(scale_crameri_sequential), 27
- scale_fill_imola
(scale_crameri_sequential), 27
- scale_fill_iridescent
(scale_tol_sequential), 45
- scale_fill_lajolla
(scale_crameri_sequential), 27
- scale_fill_land (scale_colour_land), 9
- scale_fill_lapaz
(scale_crameri_sequential), 27
- scale_fill_light (scale_tol_discrete), 39
- scale_fill_lisbon

(scale_crameri_diverging), 17
 scale_fill_muted (scale_tol_discrete),
 39
 scale_fill_nuuk
 (scale_crameri_sequential), 27
 scale_fill_okabeito
 (scale_okabeito_discrete), 38
 scale_fill_oleron
 (scale_crameri_mutlisequential),
 24
 scale_fill_oslo
 (scale_crameri_sequential), 27
 scale_fill_pale (scale_tol_discrete), 39
 scale_fill_PRGn (scale_tol_diverging),
 42
 scale_fill_roma
 (scale_crameri_diverging), 17
 scale_fill_roma0
 (scale_crameri_cyclic), 13
 scale_fill_smoothrainbow
 (scale_tol_sequential), 45
 scale_fill_soil (scale_colour_soil), 10
 scale_fill_stratigraphy
 (scale_colour_stratigraphy), 12
 scale_fill_sunset
 (scale_tol_diverging), 42
 scale_fill_tofino
 (scale_crameri_diverging), 17
 scale_fill_tokyo
 (scale_crameri_sequential), 27
 scale_fill_turku
 (scale_crameri_sequential), 27
 scale_fill_vanimo
 (scale_crameri_diverging), 17
 scale_fill_vibrant
 (scale_tol_discrete), 39
 scale_fill_vik
 (scale_crameri_diverging), 17
 scale_fill_vik0 (scale_crameri_cyclic),
 13
 scale_fill_YlOrBr
 (scale_tol_sequential), 45
 scale_okabeito_discrete, 10–12, 17, 23,
 26, 37, 38, 41, 44, 48
 scale_tol_discrete, 10–12, 17, 23, 26, 37,
 39, 39, 44, 45, 48
 scale_tol_diverging, 17, 23, 26, 37, 39, 41,
 42, 48
 scale_tol_sequential, 17, 23, 26, 37, 39,
 41, 44, 45, 45
 spacesXYZ::DeltaE(), 5