

Package ‘palettes’

July 23, 2025

Title Methods for Colour Vectors and Colour Palettes

Version 0.2.2

Description Provides a comprehensive library for colour vectors and colour palettes using a new family of colour classes (palettes_colour and palettes_palette) that always print as hex codes with colour previews. Capabilities include: formatting, casting and coercion, extraction and updating of components, plotting, colour mixing arithmetic, and colour interpolation.

License MIT + file LICENSE

URL <https://mccarthy-m-g.github.io/palettes/>,
<https://github.com/mccarthy-m-g/palettes>

BugReports <https://github.com/mccarthy-m-g/palettes/issues>

Depends R (>= 2.10)

Imports vctrs, cli, methods, pillar, rlang (>= 1.0.0), purrr,
prismatic, farver (>= 2.0.3), ggplot2 (>= 3.5.0), scales,
tibble

Suggests pkgdown, testthat (>= 3.0.0), dplyr, knitr (>= 1.22),
rmarkdown (>= 2.20), colorspace, gt (>= 0.9.0), biscale, sf,
patchwork, MetBrewer, nord, PNWColors, viridisLite, covr,
grDevices, withr

VignetteBuilder knitr

Config/Needs/website asciicast (>= 2.2.1), fontawesome

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Michael McCarthy [aut, cre, cph]

Maintainer Michael McCarthy <m.mccarthy1624@gmail.com>

Repository CRAN
Date/Publication 2025-07-01 05:30:02 UTC

Contents

as_tibble.palettes_colour	2
carto_palettes	3
colour-mixing-arithmetic	4
colour-mixing-math	5
list_colour_symbols	6
met_palettes	6
nord_palettes	7
palettes-options	8
pal_colour	9
pal_numeric	10
pal_palette	13
pal_ramp	14
penguin_palettes	16
performance_palettes	17
plot.palettes_colour	17
pnw_palettes	19
scale_colour_palette_d	20
viridis_palettes	21
Index	23

as_tibble.palettes_colour

Cast colour vectors and colour palettes to tibbles

Description

as_tibble() turns an existing colour vector or colour palette into a so-called [tibble](#), a data frame with class tbl_df.

Usage

```
## S3 method for class 'palettes_colour'
as_tibble(x, ...)

## S3 method for class 'palettes_palette'
as_tibble(x, ...)
```

Arguments

- x An object of class [palettes_palette](#) or [palettes_colour](#).
- ... Not used.

Value

A [tibble](#). The output has the following properties:

- For objects of class [palettes_colour](#): A tibble with column colour containing the colour vector.
- For objects of class [palettes_palette](#): A tibble with columns palette and colour containing palette names and colour vectors.

See Also

[pal_colour\(\)](#), [pal_palette\(\)](#)

Examples

```
x <- pal_colour(c("#663171", "#EA7428", "#0C7156"))
as_tibble(x)

y <- pal_palette(
  Egypt = c("#DD5129", "#0F7BA2", "#43B284", "#FAB255"),
  Java   = c("#663171", "#CF3A36", "#EA7428", "#E2998A", "#0C7156")
)
as_tibble(y)
```

carto_palettes

Cartography palettes

Description

Discrete, sequential, and diverging palettes created by cartographers.

Usage

discrete_palettes

sequential_palettes

diverging_palettes

Format

discrete_palettes:

An object of class `palettes_palette` with 6 colour palettes. Use `names(discrete_palettes)` to return all palette names.

sequential_palettes:

An object of class `palettes_palette` with 19 colour palettes. Use `names(sequential_palettes)` to return all palette names.

`diverging_palettes:`

An object of class `palettes_palette` with 7 colour palettes. Use `names(diverging_palettes)` to return all palette names.

Source

<https://github.com/CartoDB/CartoColor/>

See Also

`pal_palette()`, `pal_colour()`

Examples

```
# Get all palettes by name.
names(discrete_palettes)
names(sequential_palettes)
names(diverging_palettes)

# Plot all palettes.
plot(discrete_palettes)
plot(sequential_palettes)
plot(diverging_palettes)
```

colour-mixing-arithmetic

Mix colour vectors with arithmetic operators

Description

These binary operators mix colour vectors with arithmetic operators.

Usage

```
## S3 method for class 'palettes_colour'
e1 + e2
```

Arguments

`e1`, `e2` Colour vectors of class `palettes_colour`.

Value

The binary operators return colour vectors of class `palettes_colour` containing the result of the element by element operations. If involving a zero-length vector the result has length zero. Otherwise, the elements of shorter vectors are recycled as necessary. The `+` operator is for additive colour mixing.

Examples

```
x <- pal_colour("red")
y <- pal_colour("blue")
x + y
```

colour-mixing-math	<i>Mix colour vectors with math functions</i>
--------------------	---

Description

These functions mix colour vectors with math functions.

Usage

```
## S3 method for class 'palettes_colour'
sum(..., na.rm = FALSE)

## S3 method for class 'palettes_colour'
cumsum(x)
```

Arguments

...	Colour vectors of class <code>palettes_colour</code> .
na.rm	Whether to include missing values. Either TRUE or FALSE.
x	An object of class <code>palettes_colour</code> .

Value

These functions return colour vectors of class `palettes_colour`:

- `sum()` returns the sum of all the colours present in its arguments with additive colour mixing.
- `cumsum()` returns a vector whose elements are the cumulative sums of the elements of the argument with additive colour mixing.

Examples

```
x <- pal_colour(c("red", "blue"))
sum(x)

x <- pal_colour(c("red", "blue", "yellow"))
cumsum(x)
```

list_colour_symbols	<i>Symbols to use in colour previews</i>
---------------------	--

Description

List the symbols available to use in colour previews.

Usage

```
list_colour_symbols()
```

Details

By default, Unicode characters are used for symbols in colour previews in UTF-8 supported outputs. They automatically fall back to ASCII characters when the output does not support them.

To change the symbol used for colour previews, set the `palettes.print_symbol` option to a symbol name listed in `list_colour_symbols()`.

Value

This function is called for its side effects and has no return value.

See Also

[help\("palettes-options"\)](#), [cli::is_utf8_output\(\)](#)

Examples

```
list_colour_symbols()
```

met_palettes	<i>Metropolitan Museum of Art palettes</i>
--------------	--

Description

Palettes inspired by works at the Metropolitan Museum of Art in New York. Pieces selected come from various time periods, regions, and mediums.

Usage

```
met_palettes
```

```
met_palettes_a11y
```

Format

`met_palettes:`

An object of class `palettes_palette` with 56 colour palettes. Use `names(met_palettes)` to return all palette names.

`met_palettes_all:`

An object of class `palettes_palette` limited to 24 colourblind accessible palettes. All colours in each palette are distinguishable with deuteranopia, protanopia, and tritanopia. Use `names(met_palettes_all)` to return all palette names.

Author(s)

Blake Robert Mills

Source

<https://github.com/BlakeMills/MetBrewer>

See Also

`pal_palette()`, `pal_colour()`, `MetBrewer::met.brewer()`

Examples

```
# Get all palettes by name.
names(met_palettes)

# Plot all palettes.
plot(met_palettes)
```

nord_palettes

Nord palettes

Description

Dimmed pastel palettes inspired by the Arctic and Canadian wilderness.

Usage

```
nord_palettes
```

Format

`nord_palettes:`

An object of class `palettes_palette` with 16 colour palettes. Use `names(nord_palettes)` to return all palette names.

Author(s)

Jake Kaupp

Source

<https://github.com/jkaupp/nord>

See Also

[pal_palette\(\)](#), [pal_colour\(\)](#), [nord::nord\(\)](#)

Examples

```
# Get all palettes by name.
names(nord_palettes)

# Plot all palettes.
plot(nord_palettes)
```

palettes-options

Package options

Description

Options that adjust the behaviour of the palettes package.

Details

These options can be set via [options\(\)](#) and queried via [getOption\(\)](#).

Options for the palettes package

palettes.print_symbol: Character string setting the symbol used for colour previews. See [list_colour_symbols\(\)](#) for a list of symbol choices. Defaults to "circle_small". Set to FALSE to disable printing symbols.

palettes.print_hex: Logical setting whether to print hex codes in colour previews. Defaults to TRUE.

palettes.print_alpha: Logical setting whether to print the hex code alpha channel in colour previews. Defaults to FALSE. Colours without an alpha channel will be assumed to be full opacity.

palettes.print_sep: Character string to separate colours by in colour previews. Defaults to "".

palettes.print_width: Integer setting the maximum number of colours on a line in colour previews. Defaults to 1.

palettes.print_index: Logical setting whether to print the index of the first colour on each line in colour previews. Defaults to FALSE.

Note

To disable formatting in colour previews set both `palettes.print_symbol` and `palettes.print_hex` to `FALSE`.

Examples

```
options(
  palettes.print_symbol = "square",
  palettes.print_hex = FALSE,
  palettes.print_sep = " ",
  palettes.print_width = 3,
  palettes.print_index = TRUE
)
met_palettes$Cross
```

pal_colour

Colour vectors

Description

This creates a character vector that represents colours so when it is printed, colours will be formatted as hexadecimal strings.

Usage

```
pal_colour(x = character())

is_colour(x)

as_colour(x)

## Default S3 method:
as_colour(x)

## S3 method for class 'palettes_palette'
as_colour(x)
```

Arguments

- | | |
|---|---|
| x | <ul style="list-style-type: none"> • For <code>pal_colour()</code>: A character vector of any of the three kinds of R colour specifications. • For <code>as_colour()</code>: An object to be coerced. • For <code>is_colour()</code>: An object to test. |
|---|---|

Details

Colours can be specified using either:

- Hexadecimal strings of the form "#RRGGBB" or "#RRGGBBAA"
- Colour names from `grDevices::colors()`
- Positive integers `i` that index into `grDevices::palette()[i]`

Value

An S3 vector of class `palettes_colour`.

See Also

`pal_palette()`

Examples

```
pal_colour(c("darkred", "#0F7BA2"))
```

```
is_colour("darkred")
```

```
is_colour(pal_colour("darkred"))
```

```
as_colour("#0F7BA2")
```

pal_numeric	<i>Colour vector and colour palette mapping</i>
-------------	---

Description

Conveniently maps data values (numeric or factor/character) to colours according to a given colour vector or colour palette.

Usage

```
pal_numeric(
  palette,
  domain,
  na.color = "#808080",
  alpha = FALSE,
  reverse = FALSE
)
```

```
pal_bin(
  palette,
  domain,
  bins = 7,
  pretty = TRUE,
```

```

    na.color = "#808080",
    alpha = FALSE,
    reverse = FALSE,
    right = FALSE
  )

  pal_quantile(
    palette,
    domain,
    n = 4,
    probs = seq(0, 1, length.out = n + 1),
    na.color = "#808080",
    alpha = FALSE,
    reverse = FALSE,
    right = FALSE
  )

  pal_factor(
    palette,
    domain,
    levels = NULL,
    ordered = FALSE,
    na.color = "#808080",
    alpha = FALSE,
    reverse = FALSE
  )

```

Arguments

palette	An object of class <code>palettes_colour</code> or <code>palettes_colour</code> .
domain	The possible values that can be mapped. For <code>pal_numeric</code> and <code>pal_bin</code> , this can be a simple numeric range (e.g. <code>c(0, 100)</code>); <code>pal_quantile</code> needs representative numeric data; and <code>pal_factor</code> needs categorical data. If <code>NULL</code> , then whenever the resulting colour function is called, the <code>x</code> value will represent the domain. This implies that if the function is invoked multiple times, the encoding between values and colours may not be consistent; if consistency is needed, you must provide a non- <code>NULL</code> domain.
na.color	The colour to return for NA values. Note that <code>na.color = NA</code> is valid.
alpha	Whether alpha channels should be respected or ignored. If <code>TRUE</code> then colors without explicit alpha information will be treated as fully opaque.
reverse	Whether the colours in <code>palette</code> should be used in reverse order. For example, if the default order of a palette goes from blue to green, then <code>reverse = TRUE</code> will result in the colors going from green to blue.
bins	Either a numeric vector of two or more unique cut points or a single number (greater than or equal to 2) giving the number of intervals into which the domain values are to be cut.

pretty	Whether to use the function <code>pretty()</code> to generate the bins when the argument bins is a single number. When <code>pretty = TRUE</code> , the actual number of bins may not be the number of bins you specified. When <code>pretty = FALSE</code> , <code>seq()</code> is used to generate the bins and the breaks may not be "pretty".
right	parameter supplied to <code>base::cut()</code> . See Details
n	Number of equal-size quantiles desired. For more precise control, use the <code>probs</code> argument instead.
probs	See <code>stats::quantile()</code> . If provided, the <code>n</code> argument is ignored.
levels	An alternate way of specifying levels; if specified, domain is ignored
ordered	If <code>TRUE</code> and domain needs to be coerced to a factor, treat it as already in the correct order

Details

`pal_numeric` is a simple linear mapping from continuous numeric data to an interpolated palette.

`pal_bin` also maps continuous numeric data, but performs binning based on value (see the `base::cut()` function). `pal_bin` defaults for the `cut` function are `include.lowest = TRUE` and `right = FALSE`.

`pal_quantile` similarly bins numeric data, but via the `stats::quantile()` function.

`pal_factor` maps factors to colours. If the palette is discrete and has a different number of colours than the number of factors, interpolation is used.

Value

A function that takes a single parameter `x`; when called with a vector of numbers (except for `pal_factor`, which expects factors/characters), `#RRGGBB` colour strings are returned (unless `alpha = TRUE` in which case `#RRGGBBAA` may also be possible).

See Also

```
scales::col_numeric()
scales::col_bin()
scales::col_quantile()
scales::col_factor()
```

Examples

```
pal <- pal_bin(met_palettes$Tam, domain = 0:100)
plot(as_colour(pal(sort(runif(16, 0, 100)))))

# Exponential distribution, mapped continuously
pal <- pal_numeric(met_palettes$Tam, domain = NULL)
plot(as_colour(pal(sort(rexp(16)))))

# Exponential distribution, mapped by interval
pal <- pal_bin(met_palettes$Tam, domain = NULL, bins = 4)
plot(as_colour(pal(sort(rexp(16)))))
```

```
# Exponential distribution, mapped by quantile
pal <- pal_quantile(met_palettes$Tam, domain = NULL)
plot(as_colour(pal(sort(rexp(16)))))

# Categorical data; by default, the values being coloured span the gamut...
pal <- pal_factor(met_palettes$Java, domain = NULL)
plot(as_colour(pal(LETTERS[1:5])))

# ...unless the data is a factor, without droplevels...
pal <- pal_factor(met_palettes$Java, domain = NULL)
plot(as_colour(pal(factor(LETTERS[1:5], levels = LETTERS))))

# ...or the domain is stated explicitly.
pal <- pal_factor(met_palettes$Java, domain = NULL, levels = LETTERS)
plot(as_colour(pal(LETTERS[1:5])))
```

pal_palette

Colour palettes

Description

This creates a list of colour vectors.

Usage

```
pal_palette(...)
```

```
is_palette(x)
```

```
as_palette(x)
```

Arguments

- | | |
|-----|--|
| ... | <ul style="list-style-type: none"> For <code>pal_palette()</code>: A named list of character vectors of any of the three kinds of R colour specifications, or a named list of colour vectors of class <code>palettes_colour</code>. |
| x | <ul style="list-style-type: none"> For <code>as_palette()</code>: An object to be coerced. For <code>is_palette()</code>: An object to test. |

Details

Colours can be specified using either:

- Hexadecimal strings of the form `"#RRGGBB"` or `"#RRGGBBAA"`
- Colour names from `grDevices::colors()`
- Positive integers `i` that index into `grDevices::palette()[i]`

Value

An S3 list of class `palettes_palette`.

See Also

[pal_colour\(\)](#)

Examples

```
pal_palette(
  Egypt = c("#DD5129", "#0F7BA2", "#43B284", "#FAB255"),
  Java = c("#663171", "#CF3A36", "#EA7428", "#E2998A", "#0C7156")
)

x <- list(
  Egypt = c("#DD5129", "#0F7BA2", "#43B284", "#FAB255"),
  Java = c("#663171", "#CF3A36", "#EA7428", "#E2998A", "#0C7156")
)
as_palette(x)
```

pal_ramp

Colour vector and colour palette interpolation

Description

Interpolate the set of colours in [palettes_palette](#) or [palettes_colour](#) objects to create new colour palettes.

Usage

```
pal_ramp(
  palette,
  n = NULL,
  direction = 1,
  space = "lab",
  interpolate = c("linear", "spline")
)

## S3 method for class 'palettes_colour'
pal_ramp(
  palette,
  n = NULL,
  direction = 1,
  space = "lab",
  interpolate = c("linear", "spline")
)

## S3 method for class 'palettes_palette'
```

```

pal_ramp(
  palette,
  n = NULL,
  direction = 1,
  space = "lab",
  interpolate = c("linear", "spline")
)

```

Arguments

palette	An object of class <code>palettes_palette</code> or <code>palettes_colour</code> .
n	An integer specifying the number of colours to return.
direction	Sets the order of colours in the scale. If 1, the default, colours are ordered from first to last. If -1, the order of colours is reversed.
space	The colour space to interpolate in. One of: "cmy", "hsl", "hsb", "hsv", "lab" (CIE L*a*b), "hunterlab" (Hunter Lab), "oklab", "lch" (CIE Lch(ab) / polar-LAB), "luv", "rgb" (sRGB), "xyz", "yxy" (CIE xyY), "hcl" (CIE Lch(uv) / polarLuv), or "oklch" (Polar form of oklab).
interpolate	The interpolation method. Either "linear" (default) or "spline".

Value

An object of the same type as `palette`. The output has the following properties:

- For objects of class `palettes_colour`: A colour vector with `n` colours.
- For objects of class `palettes_palette`: Colour palettes with `n` colours in each palette.

See Also

`pal_colour()`, `pal_palette()`

Examples

```

# The class returned after interpolation matches the input class.
x <- pal_colour(c("darkslateblue", "cornflowerblue", "slategray1"))
y <- pal_palette(blues = x)
class(pal_ramp(x))
class(pal_ramp(y))

# Choose between linear and spline interpolation.
pal_ramp(x, n = 7, interpolate = "linear")
pal_ramp(x, n = 7, interpolate = "spline")

# Palettes will have the same length after interpolation, regardless of the
# number of colours in the original palette.
z <- pal_palette(
  Egypt = c("#DD5129", "#0F7BA2", "#43B284", "#FAB255"),
  Java = c("#663171", "#CF3A36", "#EA7428", "#E2998A", "#0C7156")
)
pal_ramp(z, n = 5)

```

penguin_palettes	<i>Palmer penguins palettes</i>
------------------	---------------------------------

Description

Palettes inspired by the Palmer penguins.

Usage

```
penguin_palettes
```

Format

```
penguin_palettes:
```

An object of class `palettes_palette` with 6 colour palettes. Use `names(penguin_palettes)` to return all palette names.

Author(s)

Allison Horst

Source

<https://github.com/allisonhorst/palmerpenguins/>

See Also

[pal_palette\(\)](#), [pal_colour\(\)](#)

Examples

```
# Get all palettes by name.
names(penguin_palettes)

# Plot all palettes.
plot(penguin_palettes)
```

performance_palettes	<i>Performance palettes</i>
----------------------	-----------------------------

Description

A colourblind accessible palette for visualizing performance.

Usage

```
performance_palettes
```

Format

```
performance_palettes:
```

An object of class `palettes_palette` with 1 colour palettes. All colours in each palette are distinguishable with deuteranopia, protanopia, and tritanopia. Use `names(performance_palettes)` to return all palette names.

Source

<https://github.com/easystats/performance/>

See Also

[pal_palette\(\)](#), [pal_colour\(\)](#)

Examples

```
# Get all palettes by name.  
names(performance_palettes)  
  
# Plot all palettes.  
plot(performance_palettes)
```

plot.palettes_colour	<i>Plot colour vectors and colour palettes</i>
----------------------	--

Description

Plot colour vectors and colour palettes as swatches.

Usage

```
## S3 method for class 'palettes_colour'
plot(
  x,
  n = NULL,
  direction = 1,
  space = "lab",
  interpolate = c("linear", "spline"),
  ...
)

## S3 method for class 'palettes_palette'
plot(
  x,
  n = NULL,
  direction = 1,
  space = "lab",
  interpolate = c("linear", "spline"),
  ...
)
```

Arguments

x	An object of class palettes_palette or palettes_colour .
n	An integer specifying the number of colours to return.
direction	Sets the order of colours in the scale. If 1, the default, colours are ordered from first to last. If -1, the order of colours is reversed.
space	The colour space to interpolate in. One of: "cmy", "hsl", "hsb", "hsv", "lab" (CIE L*ab), "hunterlab" (Hunter Lab), "oklab", "lch" (CIE Lch(ab) / polar-LAB), "luv", "rgb" (sRGB), "xyz", "yxy" (CIE xyY), "hcl" (CIE Lch(uv) / polarLuv), or "oklch" (Polar form of oklab).
interpolate	The interpolation method. Either "linear" (default) or "spline".
...	Not used.

Value

A [ggplot2](#) object. The output has the following properties:

- For objects of class [palettes_colour](#): A plot of colour swatches.
- For objects of class [palettes_palette](#) with one palette: A plot of colour swatches with the palette name spanned across the swatches.
- For objects of class [palettes_palette](#) with more than one palette: A faceted plot of colour swatches with palette names as facet titles.

See Also

[pal_colour\(\)](#), [pal_palette\(\)](#), [pal_ramp\(\)](#)

Examples

```
# Objects of class `palettes_colour` are plotted as swatches.
x <- pal_colour(c("darkslateblue", "cornflowerblue", "slategray1"))
plot(x)

# Objects of class `palettes_palette` with one palette are plotted with
# the palette name spanned across the swatches.
y <- pal_palette(Egypt = c("#DD5129", "#0F7BA2", "#43B284", "#FAB255"))
plot(y)

# Objects of class `palettes_palette` with multiple palettes are faceted.
z <- pal_palette(
  Egypt = c("#DD5129", "#0F7BA2", "#43B284", "#FAB255"),
  Java  = c("#663171", "#CF3A36", "#EA7428", "#E2998A", "#0C7156")
)
plot(z)

# Colours can also be interpolated.
plot(x, n = 5)
plot(y, n = 5)
plot(z, n = 5)
```

pnw_palettes

*Pacific Northwest palettes***Description**

Palettes inspired by Jake Lawlor's photos of the dreamiest, most colourful, PNW-iest places in Washington State.

Usage

```
pnw_palettes
```

Format

```
pnw_palettes:
```

An object of class `palettes_palette` with 14 colour palettes. Use `names(pnw_palettes)` to return all palette names.

Author(s)

Jake Lawlor

Source

<https://github.com/jakelawlor/PNWColors>

See Also

[pal_palette\(\)](#), [pal_colour\(\)](#), [PNWColors::pnw_palette\(\)](#)

Examples

```
# Get all palettes by name.
names(pnw_palettes)

# Plot all palettes.
plot(pnw_palettes)
```

scale_colour_palette_d

Colour scales from colour vectors and colour palettes

Description

Create discrete, continuous, and binned colour scales from colour vectors and colour palettes.

Usage

```
scale_colour_palette_d(palette, direction = 1, ...)

scale_fill_palette_d(palette, direction = 1, ...)

scale_colour_palette_c(palette, direction = 1, ..., na.value = NA)

scale_fill_palette_c(palette, direction = 1, ..., na.value = NA)

scale_colour_palette_b(palette, direction = 1, ...)

scale_fill_palette_b(palette, direction = 1, ...)
```

Arguments

palette	An object of class palettes_palette or palettes_colour .
direction	Sets the order of colours in the scale. If 1, the default, colours are ordered from first to last. If -1, the order of colours is reversed.
...	Other arguments passed on to ggplot2::discrete_scale() , ggplot2::continuous_scale() , or ggplot2::binned_scale() to control name, limits, breaks, labels and so forth.
na.value	Missing values will be replaced with this value.

Value

A scale function that controls the mapping between data and colour or fill aesthetics in a [ggplot2](#) plot.

Examples

```
library(ggplot2)

# Use palette_d with discrete data
discrete_pal <- pal_colour(c("#663171", "#EA7428", "#0C7156"))
ggplot(mtcars, aes(wt, mpg, colour = as.factor(cyl))) +
  geom_point(size = 3) +
  scale_colour_palette_d(discrete_pal)

# Use palette_c with continuous data
continuous_pal <- pal_colour(c("#3C0D03", "#E67424", "#F5C34D"))
ggplot(mtcars, aes(wt, mpg, colour = mpg)) +
  geom_point(size = 3) +
  scale_colour_palette_c(continuous_pal)

# Use palette_b to bin continuous data before mapping
ggplot(mtcars, aes(wt, mpg, colour = mpg)) +
  geom_point(size = 3) +
  scale_colour_palette_b(continuous_pal)
```

viridis_palettes

Viridis palettes

Description

Colourblind accessible palettes that are perceptually uniform in both colour and black-and-white.

Usage

```
viridis_palettes
```

Format

viridis_palettes:

An object of class `palettes_palette` with 8 colour palettes. All colours in each palette are distinguishable with deuteranopia, protanopia, and tritanopia. Use `names(viridis_palettes)` to return all palette names.

Author(s)

Simon Garnier

Source

<https://github.com/sjmgarnier/viridisLite>

See Also

[pal_palette\(\)](#), [pal_colour\(\)](#), [viridisLite::viridis\(\)](#)

Examples

```
# Get all palettes by name.  
names(viridis_palettes)  
  
# Plot all palettes.  
plot(viridis_palettes, n = 256)
```

Index

- * **datasets**
 - carto_palettes, 3
 - met_palettes, 6
 - nord_palettes, 7
 - penguin_palettes, 16
 - performance_palettes, 17
 - pnw_palettes, 19
 - viridis_palettes, 21
- +.palettes_colour
 - (colour-mixing-arithmetic), 4
- as_color (pal_colour), 9
- as_colour (pal_colour), 9
- as_palette (pal_palette), 13
- as_tibble.palettes_colour, 2
- as_tibble.palettes_palette
 - (as_tibble.palettes_colour), 2
- base::cut(), 12
- carto_palettes, 3
- cli::is_utf8_output(), 6
- color-mixing-arithmetic
 - (colour-mixing-arithmetic), 4
- color-mixing-math (colour-mixing-math), 5
- colour-mixing-arithmetic, 4
- colour-mixing-math, 5
- cumsum.palettes_colour
 - (colour-mixing-math), 5
- discrete_palettes (carto_palettes), 3
- discrete_palettes, (carto_palettes), 3
- diverging_palettes (carto_palettes), 3
- getOption(), 8
- ggplot2, 18, 20
- ggplot2::binned_scale(), 20
- ggplot2::continuous_scale(), 20
- ggplot2::discrete_scale(), 20
- grDevices::colors(), 10, 13
- grDevices::palette(), 10, 13
- is_color (pal_colour), 9
- is_colour (pal_colour), 9
- is_palette (pal_palette), 13
- list_color_symbols
 - (list_colour_symbols), 6
- list_colour_symbols, 6
- list_colour_symbols(), 8
- met_palettes, 6
- met_palettes_a11y (met_palettes), 6
- MetBrewer::met.brewer(), 7
- nord::nord(), 8
- nord_palettes, 7
- options(), 8
- pal_bin (pal_numeric), 10
- pal_color (pal_colour), 9
- pal_colour, 9
- pal_colour(), 3, 4, 7, 8, 14–18, 20, 21
- pal_factor (pal_numeric), 10
- pal_numeric, 10
- pal_palette, 13
- pal_palette(), 3, 4, 7, 8, 10, 15–18, 20, 21
- pal_quantile (pal_numeric), 10
- pal_ramp, 14
- pal_ramp(), 18
- palettes-options, 8
- palettes_colour, 2–5, 11, 13–15, 18, 20
- palettes_palette, 2, 3, 14, 15, 18, 20
- penguin_palettes, 16
- performance_palettes, 17
- plot.palettes_colour, 17
- plot.palettes_palette
 - (plot.palettes_colour), 17
- pnw_palettes, 19
- PNWColors::pnw_palette(), 20

`pretty()`, [12](#)

`scale_color_palette_b`
 (`scale_colour_palette_d`), [20](#)

`scale_color_palette_c`
 (`scale_colour_palette_d`), [20](#)

`scale_color_palette_d`
 (`scale_colour_palette_d`), [20](#)

`scale_colour_palette_b`
 (`scale_colour_palette_d`), [20](#)

`scale_colour_palette_c`
 (`scale_colour_palette_d`), [20](#)

`scale_colour_palette_d`, [20](#)

`scale_fill_palette_b`
 (`scale_colour_palette_d`), [20](#)

`scale_fill_palette_c`
 (`scale_colour_palette_d`), [20](#)

`scale_fill_palette_d`
 (`scale_colour_palette_d`), [20](#)

`scales::col_bin()`, [12](#)

`scales::col_factor()`, [12](#)

`scales::col_numeric()`, [12](#)

`scales::col_quantile()`, [12](#)

`seq()`, [12](#)

`sequential_palettes` (`carto_palettes`), [3](#)

`sequential_palettes`, (`carto_palettes`), [3](#)

`stats::quantile()`, [12](#)

`sum.palettes_colour`
 (`colour-mixing-math`), [5](#)

`tibble`, [2](#), [3](#)

`viridis_palettes`, [21](#)

`viridisLite::viridis()`, [21](#)