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

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

```r
## 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.
Examples

```r
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)
```

---

**colour-mixing-arithmetic**

*Mix colour vectors with arithmetic operators*

---

**Description**

These binary operators mix colour vectors with arithmetic operators.

**Usage**

```r
## 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**

```r
x <- pal_colour("red")
y <- pal_colour("blue")
x + y
```
colour-mixing-math  Mix colour vectors with math functions

Description
These functions mix colour vectors with math functions.

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

Arguments
- `...`: Colour vectors of class `palettes_colour`.
- `na.rm`: Whether to include missing values. Either `TRUE` or `FALSE`.
- `x`: An object of class `palettes_colour`.

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
```r
x <- pal_colour(c("red", "blue"))
sum(x)
```
```r
x <- pal_colour(c("red", "blue", "yellow"))
cumsum(x)
```
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_a11y:
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_a11y) to return all palette names.
```

Author(s)

Blake Robert Mills

Source

https://github.com/BlakeRMills/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)
```
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)
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 • For pal_colour(): A character vector of any of the three kinds of R colour specifications.
   • For as_colour(): An object to be coerced.
   • For is_colour(): 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

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 palettes_palette or palettes_colour.
domain        The possible values that can be mapped.
For pal_numeric and pal_bin, this can be a simple numeric range (e.g. c(0, 100)); pal_quantile needs representative numeric data; and pal_factor needs categorical data.
If NULL, then whenever the resulting colour function is called, the x 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-NULL domain.

na.color      The colour to return for NA values. Note that na.color = NA is valid.
alpha         Whether alpha channels should be respected or ignored. If TRUE then colors without explicit alpha information will be treated as fully opaque.
reverse       Whether the colours in palette should be used in reverse order. For example, if the default order of a palette goes from blue to green, then reverse = TRUE 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 pretty() to generate the bins when the argument bins is a single number. When pretty = TRUE, the actual number of bins may not be the number of bins you specified. When pretty = FALSE, seq() is used to generate the bins and the breaks may not be "pretty".

right         parameter supplied to base::cut(). See Details

n             Number of equal-size quantiles desired. For more precise control, use the probs argument instead.
probs         See stats::quantile(). If provided, the n argument is ignored.
levels        An alternate way of specifying levels; if specified, domain is ignored
ordered       If TRUE and domain needs to be coerced to a factor, treat it as already in the correct order
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
... • For pal_palette(): 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 palettes_colour.
x • For as_palette(): An object to be coerced.
 • For is_palette(): 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")
)
Java = c("663171", "CF3A36", "EA7428", "E2998A", "C7156")

as_palette(x)

descriptions: 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 palettes_palette or palettesColour.
- **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".

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.

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)

plot.palettes_colour
Plot colour vectors and colour palettes

Description
Plots a colour palette object.

Usage

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

plot.palettes_colour
plot.palettes_colour

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", "hs1", "hsb", "hsv", "lab" (CIE L*ab), "hunterlab" (Hunter Lab), "oklab", "lch" (CIE Lch(ab) / polar-LAB), "lue", "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_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

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

Arguments

palette

direction

... Other arguments passed on to ggplot2::discrete_scale(), ggplot2::continuous_scale(), or ggplot2::binned_scale() to control name, limits, breaks, labels and so forth.

Value

A scale function that controls the mapping between data and colour or fill aesthetics in a ggplot2 plot.
**Examples**

```r
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](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
  met_palettes, 5
  nord_palettes, 6
  pnw_palettes, 15
  viridis_palettes, 17
+.palettes_colour
  (colour-mixing-arithmetic), 3
as_color (pal_colour), 7
as_colour (pal_colour), 7
as_palette (pal_palette), 11
as_tibble.palettes_colour, 2
as_tibble.palettes_palette
  (as_tibble.palettes_colour), 2
base::cut(), 9, 10
color-mixing-arithmetic
  (colour-mixing-arithmetic), 3
color-mixing-math (colour-mixing-math), 4
colour-mixing-arithmetic, 3
colour-mixing-math, 4
cumsum.palettes_colour
  (colour-mixing-math), 4
ggplot2, 14, 16
is_color (pal_colour), 7
is_colour (pal_colour), 7
is_palette (pal_palette), 11
met_palettes, 5
met_palette_all (met_palettes), 5
MetBrewer::met.brewer(), 5
nord::nord(), 6
nord_palettes, 6
pal_bin (pal_numeric), 8
pal_color (pal_colour), 7
pal_colour, 7
pal_colour(), 5, 6, 15, 17
pal_factor (pal_numeric), 8
pal_numeric, 8
pal_palette, 11
pal_palette(), 5, 6, 15, 17
pal_quantile (pal_numeric), 8
pal_ramp, 12
plot.palettes_colour, 13
plot.palettes_palette
  (plot.palettes_colour), 13
pnw_palettes, 15
PNWColors::pnw_palette(), 15
pretty(), 9
scale_color_palette_b
  (scale_colour_palette_d), 16
scale_color_palette_c
  (scale_colour_palette_d), 16
scale_color_palette_d
  (scale_colour_palette_d), 16
scale_colour_palette_b
  (scale_colour_palette_d), 16
scale_colour_palette_c
  (scale_colour_palette_d), 16
scale_colour_palette_d
  (scale_colour_palette_d), 16
scale_fill_palette_b
  (scale_colour_palette_d), 16
scale_fill_palette_c
  (scale_colour_palette_d), 16
scale_fill_palette_d
  (scale_colour_palette_d), 16
scales::col_bin(), 10
scales::col_factor(), 10
scales::col_numeric(), 10
scales::col_quantile(), 10
seq(), 9
stats::quantile(), 9, 10
sum.palettes_colour
  (colour-mixing-math), 4
tibble, 2

viridis_palettes, 17
viridisLite::viridis(), 17