Package 'colorify'

May 8, 2026

Type Package

Title Intuitive Color and Palette Generation and Modification

Version 0.1.2

Description A one-stop shop for intuitive and dependency-free color and palette creation and modification. Includes palettes and functionality from popular packages such as 'viridis', 'RColorBrewer', and base R 'grDevices', as well as 'ggplot2' plot bindings. Users can generate perceptually uniform and colorblind-friendly palettes, adjust palettes in HSL and RGB color spaces, map color gradients to value ranges, and create color-generating functions.

URL https://github.com/mauritsunkel/colorify, https://mauritsunkel.github.io/colorify/

BugReports https://github.com/mauritsunkel/colorify/issues

License Apache License 2.0

Encoding UTF-8

RoxygenNote 7.3.3

Config/testthat/edition 3

Depends R (>= 4.3)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), ggplot2, circlize

VignetteBuilder knitr

NeedsCompilation no

Author Maurits Unkel [aut, cre]

Maintainer Maurits Unkel <mauritsunkel@gmail.com>

Repository CRAN

Date/Publication 2025-11-06 10:10:07 UTC

Contents

`colorify` *colorify: creation and modification of color/gradient palettes*

Description

The main colorify function can be used to generate or take colors that can then be modified with the same function call. See the vignette for extended examples.

Palette names are stripped of whitespace and lowered for name matching. All RColorBrewer and Viridis palettes are included. All grDevices plotting functions are provided as palettes, simply use: colors = "rainbow", "heat", "terrain", "topo" or "cm". Viridis is recommended for (continuous) color-blind friendly paletets. Okabe-Ito is recommended for discrete distinct colors (up to 8, generate if more colors are required).

Addition of values (.v) happens before multiplication with factors (.f). Intuitively, all given values are expected to be within range (0, 100), values will be scaled between (0, 1), as hsv() and rgb2hsv(maxColorValue = 1) require.

Note that parameter call order within the function call matters, see examples and vignette.

A wrapper function around `colorify` to turn it into a palette function compatible with `discrete_scale` and `scale_fill_gradientn`.

Usage

```
colorify(
  n = NULL,
  colors = character(0),
  colours = colors,
  colors_lock = NULL,
  colors_names = character(0),
  colors_map = numeric(0),
  nn = n,
  hf = 1,
  sf = 1,
  lf = 1,
  rf = 1,
  gf = 1,
  bf = 1,
  hv = 0,
  sv = 0,
  lv = 0,
```

```
  rv = 0L,
  gv = 0L,
  bv = 0L,
  hmin = 0L,
  smin = 0L,
  lmin = 0L,
  rmin = 0L,
  gmin = 0L,
  bmin = 0L,
  hmax = 100L,
  smax = 100L,
  lmax = 100L,
  rmax = 100L,
  gmax = 100L,
  bmax = 100L,
  alpha = 1,
  seed = 42L,
  order = 1,
  plot = FALSE,
  export = FALSE,
  verbose = TRUE,
  ...
)

colorify_pal(
  colors = character(0),
  colors_lock = NULL,
  hf = 1,
  sf = 1,
  lf = 1,
  rf = 1,
  gf = 1,
  bf = 1,
  hv = 0,
  sv = 0,
  lv = 0,
  rv = 0L,
  gv = 0L,
  bv = 0L,
  hmin = 0L,
  smin = 0L,
  lmin = 0L,
  rmin = 0L,
  gmin = 0L,
  bmin = 0L,
  hmax = 100L,
  smax = 100L,
  lmax = 100L,
```

```
  rmax = 100L,
  gmax = 100L,
  bmax = 100L,
  alpha = 1,
  seed = 42L,
  order = 1,
  verbose = TRUE,
  ...
)
```

Arguments

n	integer, default: NULL, else amount of colors to get, if palette selected and more colors requested they will be generated
colors	character (vector), combination of selecting palette(s) by name (options: see display_palettes()), and/or vector of R color names and/or color hexcodes
colours	colors
colors_lock	numeric/boolean, default: NULL, numerical or logical index of colors (not) to be modified, if logical length != colors it will be cut or filled with TRUE/FALSE, prefix with '!' for logical vectors and '-' for numerical vectors to get inverse, see examples. If nn %% length(colors) == 0, i.e. if nn divisive by amount of colors without rest, set repeat given locking pattern
colors_names	character, default: character(0), else return named vector of final colors
colors_map	numeric, default numeric(0), else vector of n values for colors to make gradient map between and return function
nn	integer (vector), default: n, else amount(s) of colors to output as gradient(s), after completing palette for n colors, if Inf return a callable function(n) generating colors
hf	hue factor, default: 1, multiply values by factor, proportional to base value of 1
sf	saturation factor, default: 1, multiply values by factor, proportional to base value of 1
lf	lightness/brightness factor, default: 1, multiply values by factor, proportional to base value of 1
rf	red factor, default: 1, multiply values by factor, proportional to base value of 1
gf	green factor, default: 1, multiply values by factor, proportional to base value of 1
bf	blue factor, default: 1, multiply values by factor, proportional to base value of 1
hv	hue value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
sv	saturation value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
lv	lightness/brightness value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100

rv	red value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
gv	green value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
bv	blue value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
hmin	hue minimum threshold, default: 0, expected range (0, 100)
smin	saturation minimum threshold, default: 0, expected range (0, 100)
lmin	lightness/brightness minimum threshold, default: 0, expected range (0, 100)
rmin	red minimum threshold, default: 0, expected range (0, 100)
gmin	green minimum threshold, default: 0, expected range (0, 100)
bmin	blue minimum threshold, default: 0, expected range (0, 100)
hmax	hue maximum threshold, default: 0, expected range (0, 100)
smax	saturation maximum threshold, default: 0, expected range (0, 100)
lmax	lightness/brightness maximum threshold, default: 0, expected range (0, 100)
rmax	red maximum threshold, default: 0, expected range (0, 100)
gmax	green maximum threshold, default: 0, expected range (0, 100)
bmax	blue maximum threshold, default: 0, expected range (0, 100)
alpha	numeric, sets color alpha values
seed	integer, default: 42, set seed for generation of colors (n > given colors (palettes)) and colors ordering (see order)
order	default: 1, numeric (vector) to adjust colors order, -1: reverse order, 0: seeded random order, >1: shift order, c(-1, >1): reverse then shift order, or numeric vector as many colors to set custom order (if longer, vector shortened to n colors)
plot	default: FALSE, if TRUE or string, plot pie chart of color palette, if 'i' in string then plot image instead of pie, if 'l' in string plot color index as labels
export	default: FALSE, if TRUE: export = getwd(), if export = "string/", save hexcodes, rgb, and hsl values to export/colorify.csv
verbose	default: TRUE, mentions if and how many colors are generated
...	Use the ellipsis parameter to set color space and interpolate for grDevices::colorRampPalette()

Details

Either generate theoretically maximally different colors, select an available R grDevices palette and/or modify the colors of the given gradient/palette

Value

colorify: vector of color hexcodes

colorify_pal: callable function(n) for generating colors

See Also

vignette("Introduction to coloRify")

Browse vignettes with vignette("Introduction to coloRify")

Examples

```
## if parameters identical, change seed to change generation
colorify(10, plot = TRUE, seed = 1)
colorify(10, plot = TRUE, seed = 42)
## set colors, generate additional up to n
colorify(colors = c("red", "white", "blue"), n = 5, plot = TRUE)
## create gradients
colorify(colors = c("orange", "red", "white", "blue", "orange"), nn = 100, plot = TRUE)

## viridis gradient, lighten and saturate, darken
colorify(colors = "viridis", n = 100, plot = TRUE)
colorify(colors = "viridis", n = 10, plot = TRUE, lf = 1.5, sv = 10)
colorify(colors = "viridis", n = 10, plot = TRUE, lf = .9)
# TODO add examples for nn to vignette
colorify(colors = colorify(nn = Inf)(10), plot = TRUE) # basically random palette function
colorify(colors = colorify(nn = Inf, colors = c('red', 'white'))(10), plot = TRUE)
colorify(colors = colorify(nn = Inf, colors = c('red', 'white', 'blue'))(10), plot = TRUE)
colorify(colors = colorify(colors = 'viridis', nn = Inf)(50), plot = TRUE)

## palette selected by name in colors[1],
## can add colors to selected palette,
## if n < length, remove colors , if greater generate
colorify(colors = c("Okabe-Ito", "red", "blue", "yellow"), plot = TRUE, n = 10)

## no adjustments to locked indices
colorify(colors = "Okabe-Ito", colors_lock = c(FALSE,FALSE,TRUE,TRUE), plot = TRUE, rv = -300)
colorify(colors = "Okabe-Ito", colors_lock = c(FALSE,FALSE,TRUE,TRUE), plot = TRUE, rv = 300)

## colors_lock and inversing
colors <- colorify(5, plot = TRUE)
colorify(colors_lock = c(TRUE,TRUE), colors=colors, plot = TRUE, lf = .5)
colorify(colors_lock = ! c(TRUE,FALSE,TRUE), colors=colors, plot = TRUE, lf = .5)
colorify(colors_lock = c(3,4), colors=colors, plot = TRUE, lf = .5)
colorify(colors_lock = -c(3,4), colors=colors, plot = TRUE, lf = .5)

## rainbow
colorify(colors=grDevices::rainbow(100, s = .5), plot = TRUE)
colorify(colors="rainbow", n = 100, sf = .5, plot = TRUE)
colorify(colors=grDevices::rainbow(100, v = .5), plot = TRUE)
colorify(colors="rainbow", n = 100, lf = .5, plot = TRUE)
colorify(colors=grDevices::rainbow(100, start = .25, end = .75), plot = TRUE)
colorify(colors=grDevices::rainbow(100)[25:75], plot = TRUE)

## order
colorify(10, plot = TRUE, order = 1)  # default
colorify(10, plot = TRUE, order = 0)  # random
colorify(10, plot = TRUE, order = -1) # reverse
```

```
colorify(10, plot = TRUE, order = -3) # negative shift
colorify(10, plot = TRUE, order = 12) # > n

## call order
# Note that parameter call order within the function call matters,
# see examples and vignette:
# rv = 20 then rf = 1.2 can be different then
# rf = 1.2 then rv = 20

# TODO add example to vignette
colors_map <- c(-5, 0, 10)
colors <- c("red", "white", "blue")
if (requireNamespace('circlize'))
color_bar <-
  circlize::colorRamp2(colors_map, colors)(seq(-5, 10, length.out = 100))
color_bar <- colorify(
  colors = colors,
  colors_map = colors_map,
  space = "Lab")(seq(-5, 10, length.out = 100)) # circlize::colorRamp2 style
graphics::image(1:100, 1, as.matrix(1:100),
  col = color_bar,
  axes = FALSE,
  main = "Color Mapping using circlize::colorRamp2")
## named base R/hexcode
colorifunction <- colorify_pal(colors = c('red', '#FFFFFF', 'blue'))
colorify(colors = colorifunction(10), plot = TRUE)
## empty for random
colorifunction <- colorify_pal()
colorify(colors = colorifunction(10), plot = TRUE)
## named colors and palette(s)
colorifunction <- colorify_pal(colors = c('green', 'viridis', 'rainbow', 'yellow'), plot = TRUE)
colorify(colors = colorifunction(100), plot = TRUE)
```

`colortistry` *Colortistry plot from Colorified palettes*

Description

Colortistry plot from Colorified palettes

Usage

```
colortistry(colors_list, border_color = NA)
```

Arguments

`colors_list`	list of colors generated with `colorify`, see example
`border_color`	default: NA, for no border, otherwise R grDevices color or hexcolor

Value

plot Colortistry plot of combined Colorified palettes

See Also

Browse vignettes with `vignette("Introduction to coloRify")`

Examples

```
colors_list <- list()
for (i in seq(100)) {
  colors_list[[i]] <- colorify(
  n = 100,
  colors = "rainbow",
  colors_lock = rep(c(TRUE,FALSE,FALSE,FALSE,FALSE), 20),
  hf = 25/i,
  lf = i/20)
  if (i %% 3) colors_list[[i]] <- colorify(
  n = 100,
  colors = "rainbow",
  colors_lock = rep(c(FALSE,FALSE,TRUE,FALSE,FALSE), 20),
  hf = 30/i,
  lf = i/20)
  if (i %% 4) colors_list[[i]] <- colorify(
  n = 100,
  colors = "rainbow",
  colors_lock = rep(c(FALSE,FALSE,FALSE,TRUE,FALSE), 20),
  hf = 50/i,
  lf = i/40)
  if (i %% 5) colors_list[[i]] <- colorify(
  n = 100,
  colors = "rainbow",
  colors_lock = rep(c(FALSE,TRUE,FALSE,FALSE,FALSE), 20),
  hf = 50/i,
  sf = i/50)
}
colortistry(colors_list)
colortistry(colors_list, border_color = 'black')
```

`display_palettes` *Display R grDevices palettes*

Description

Use `colorify` to select and modify the palettes. Note that discrete palettes with maximum n colors will be repeated in plotting.

Any numeric i_palettes over maximum amount of palettes are not displayed.

Contains all Viridis palettes, including Turbo.

Usage

```
display_palettes(n = 10, i_palettes = 1:1000, border = FALSE)
```

Arguments

n	integer, amount of colors to display
i_palettes	default: numeric vector as index/range for choosing palettes, or a combination of 'rcolorbrewer', 'viridis', 'rainbow' (grDevices Palettes) to show specific palettes
border	default: FALSE, if TRUE show color rectangle borders

Value

named vector with source and name of palettes, 'hcl' for grDevices::hcl.pals() and 'pal' for grDevices::palette.pals()

See Also

Browse vignettes with `vignette("Introduction to coloRify")`

Examples

```
display_palettes()
display_palettes(i_palettes = 50:75)

display_palettes(i_palettes = 'RColorBrewer')
display_palettes(i_palettes = 'Viridis')
display_palettes(i_palettes = c("rainbow", "viridis"))

display_palettes(i_palettes = c(1,5,10,20,40,100,119))
display_palettes(n = 100, i_palettes = 1:10)
display_palettes(n = 10, i_palettes = 1:10, border = TRUE)
```

---

hex2rgba *Hex code colors to rgba format*

Description

Hex code colors to rgba format

Usage

```
hex2rgba(hex, alpha = NULL)
```

Arguments

hex	character (vector), hexcode colors (e.g. #FFFFFF)
alpha	numeric in range (0-1), default: NULL to use full opacity or given opacity (AA) in hex (#RRGGBBAA)

Value

colors in rgba format

Examples

```
colors <- colorify(5)
hex2rgba(colors)
hex2rgba(colors, alpha = .5)
colors <- gsub('FF$', 75, colors)
hex2rgba(colors)
hex2rgba(colors, alpha = .5)
```

---

hsv2rgb *hsv to rgb color space*

Description

Expects hsv color values to be in range (0-100])

Usage

```
hsv2rgb(h, s, v, maxColorValue = 100)
```

Arguments

h	numeric, vector of 'hue' values
s	numeric, vector of 'saturation' values
v	numeric, vector of 'value' (lightness) values
maxColorValue	numeric, default: 100, gives the maximum hsv color values range. Default corresponds to the typical 0:1 HSV coding as in `rgb2hsv()`

Value

unnamed dataframe with rgb colors

Examples

```
colors <- colorify(5)
rgb <- grDevices::col2rgb(colors)
hsv <- grDevices::rgb2hsv(rgb, maxColorValue = 255)
rgb2 <- hsv2rgb(hsv['h',], hsv['s',], hsv['v',], maxColorValue = 255)
```

order_by_shift *Shift colors order*

Description

Shift colors order

Usage

```
order_by_shift(shift, colors, n)
```

Arguments

shift	integer to shift colors order by
colors	hexcolors vector
n	length(colors)

Value

ordered colors vector

scale_color_colorify *coloRify scale color bindings for ggplot2*

Description

for rest and default coloRify parameters see colorify

Usage

```
scale_color_colorify(
  ...,
  aesthetics = "color",
  discrete = FALSE,
  nn = Inf,
  n = 2,
  colors = character(0),
  colors_lock = NULL,
  hf = 1,
  sf = 1,
  lf = 1,
  rf = 1,
  gf = 1,
  bf = 1,
  hv = 0,
```

```
  sv = 0,
  lv = 0,
  rv = 0,
  gv = 0,
  bv = 0,
  hmin = 0,
  smin = 0,
  lmin = 0,
  rmin = 0,
  gmin = 0,
  bmin = 0,
  hmax = 100,
  smax = 100,
  lmax = 100,
  rmax = 100,
  gmax = 100,
  bmax = 100,
  alpha = 1,
  seed = 42,
  order = 1,
  verbose = TRUE
)

scale_colour_colorify(
  ...,
  aesthetics = "color",
  discrete = FALSE,
  nn = Inf,
  n = 2,
  colors = character(0),
  colors_lock = NULL,
  hf = 1,
  sf = 1,
  lf = 1,
  rf = 1,
  gf = 1,
  bf = 1,
  hv = 0,
  sv = 0,
  lv = 0,
  rv = 0,
  gv = 0,
  bv = 0,
  hmin = 0,
  smin = 0,
  lmin = 0,
  rmin = 0,
  gmin = 0,
```

```
  bmin = 0,
  hmax = 100,
  smax = 100,
  lmax = 100,
  rmax = 100,
  gmax = 100,
  bmax = 100,
  alpha = 1,
  seed = 42,
  order = 1,
  verbose = TRUE
)
```

Arguments

...	additional parameters passed to colorify, discrete_scale and/or scale_color_gradientn
aesthetics	string, default: 'color', see discrete_scale and scale_color_gradientn for more aesthetics
discrete	boolean, default = FALSE (calls scale_color_gradientn), else TRUE (calls discrete_scale)
nn	integer, default: Inf, length of gradients, if Inf then set to 256
n	integer, default: 2, amount of colors(/gradients) to return, only works if not discrete
colors	character (vector), combination of selecting palette(s) by name (options: see display_palettes()), and/or vector of R color names and/or color hexcodes
colors_lock	numeric/boolean, default: NULL, numerical or logical index of colors (not) to be modified, if logical length != colors it will be cut or filled with TRUE/FALSE, prefix with '!' for logical vectors and '-' for numerical vectors to get inverse, see examples. If nn %% length(colors) == 0, i.e. if nn divisive by amount of colors without rest, set repeat given locking pattern
hf	hue factor, default: 1, multiply values by factor, proportional to base value of 1
sf	saturation factor, default: 1, multiply values by factor, proportional to base value of 1
lf	lightness/brightness factor, default: 1, multiply values by factor, proportional to base value of 1
rf	red factor, default: 1, multiply values by factor, proportional to base value of 1
gf	green factor, default: 1, multiply values by factor, proportional to base value of 1
bf	blue factor, default: 1, multiply values by factor, proportional to base value of 1
hv	hue value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
sv	saturation value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
lv	lightness/brightness value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100

rv	red value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
gv	green value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
bv	blue value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
hmin	hue minimum threshold, default: 0, expected range (0, 100)
smin	saturation minimum threshold, default: 0, expected range (0, 100)
lmin	lightness/brightness minimum threshold, default: 0, expected range (0, 100)
rmin	red minimum threshold, default: 0, expected range (0, 100)
gmin	green minimum threshold, default: 0, expected range (0, 100)
bmin	blue minimum threshold, default: 0, expected range (0, 100)
hmax	hue maximum threshold, default: 0, expected range (0, 100)
smax	saturation maximum threshold, default: 0, expected range (0, 100)
lmax	lightness/brightness maximum threshold, default: 0, expected range (0, 100)
rmax	red maximum threshold, default: 0, expected range (0, 100)
gmax	green maximum threshold, default: 0, expected range (0, 100)
bmax	blue maximum threshold, default: 0, expected range (0, 100)
alpha	numeric, sets color alpha values
seed	integer, default: 42, set seed for generation of colors (n > given colors (palettes)) and colors ordering (see order)
order	default: 1, numeric (vector) to adjust colors order, -1: reverse order, 0: seeded random order, >1: shift order, c(-1, >1): reverse then shift order, or numeric vector as many colors to set custom order (if longer, vector shortened to n colors)
verbose	default: TRUE, mentions if and how many colors are generated

Value

sets colors in ggplot2 plotted object, see examples

See Also

`vignette("Introduction to coloRify")`

Examples

```
## viridis ggplot2 examples Colorified

## non-discrete
if (requireNamespace("ggplot2", quietly = TRUE)) {
  dsub <- subset(ggplot2::diamonds, x > 5 & x < 6 & y > 5 & y < 6)
  dsub$diff <- with(dsub, sqrt(abs(x - y)) * sign(x - y))

  d <- ggplot2::ggplot(dsub, ggplot2::aes(x, y, colour = diff)) + ggplot2::geom_point()
```

```
    d + scale_color_colorify(n = 4, colors = 'viridis') + ggplot2::theme_bw()


## discrete

  p <- ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg))
  p + ggplot2::geom_point(size = 4, ggplot2::aes(colour = factor(cyl))) +
   ggplot2::theme_bw() + scale_color_colorify(discrete = TRUE, colors = c('red', 'blue', 'yellow'))
}
```

---

scale_fill_colorify *coloRify scale fill bindings for ggplot2*

Description

for rest and default coloRify parameters see colorify

Usage

```
scale_fill_colorify(
  ...,
  aesthetics = "fill",
  discrete = FALSE,
  nn = Inf,
  n = 2,
  colors = character(0),
  colors_lock = NULL,
  hf = 1,
  sf = 1,
  lf = 1,
  rf = 1,
  gf = 1,
  bf = 1,
  hv = 0,
  sv = 0,
  lv = 0,
  rv = 0,
  gv = 0,
  bv = 0,
  hmin = 0,
  smin = 0,
  lmin = 0,
  rmin = 0,
  gmin = 0,
  bmin = 0,
  hmax = 100,
  smax = 100,
  lmax = 100,
```

```
  rmax = 100,
  gmax = 100,
  bmax = 100,
  alpha = 1,
  seed = 42,
  order = 1,
  verbose = TRUE
)
```

Arguments

...	additional parameters passed to colorify, discrete_scale and/or scale_fill_gradientn
aesthetics	string, default: 'fill', see discrete_scale and scale_fill_gradientn for more aesthetics
discrete	boolean, default = FALSE (calls scale_fill_gradientn), else TRUE (calls discrete_scale)
nn	integer, default: Inf, length of gradients, if Inf then set to 256
n	integer, default: 2, amount of colors(/gradients) to return, only works if not discrete
colors	character (vector), combination of selecting palette(s) by name (options: see display_palettes()), and/or vector of R color names and/or color hexcodes
colors_lock	numeric/boolean, default: NULL, numerical or logical index of colors (not) to be modified, if logical length != colors it will be cut or filled with TRUE/FALSE, prefix with '!' for logical vectors and '-' for numerical vectors to get inverse, see examples. If nn %% length(colors) == 0, i.e. if nn divisive by amount of colors without rest, set repeat given locking pattern
hf	hue factor, default: 1, multiply values by factor, proportional to base value of 1
sf	saturation factor, default: 1, multiply values by factor, proportional to base value of 1
lf	lightness/brightness factor, default: 1, multiply values by factor, proportional to base value of 1
rf	red factor, default: 1, multiply values by factor, proportional to base value of 1
gf	green factor, default: 1, multiply values by factor, proportional to base value of 1
bf	blue factor, default: 1, multiply values by factor, proportional to base value of 1
hv	hue value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
sv	saturation value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
lv	lightness/brightness value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
rv	red value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100

gv	green value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
bv	blue value, default: 0, add value to values, linear from base value of 0 to a maximum value of 100
hmin	hue minimum threshold, default: 0, expected range (0, 100)
smin	saturation minimum threshold, default: 0, expected range (0, 100)
lmin	lightness/brightness minimum threshold, default: 0, expected range (0, 100)
rmin	red minimum threshold, default: 0, expected range (0, 100)
gmin	green minimum threshold, default: 0, expected range (0, 100)
bmin	blue minimum threshold, default: 0, expected range (0, 100)
hmax	hue maximum threshold, default: 0, expected range (0, 100)
smax	saturation maximum threshold, default: 0, expected range (0, 100)
lmax	lightness/brightness maximum threshold, default: 0, expected range (0, 100)
rmax	red maximum threshold, default: 0, expected range (0, 100)
gmax	green maximum threshold, default: 0, expected range (0, 100)
bmax	blue maximum threshold, default: 0, expected range (0, 100)
alpha	numeric, sets color alpha values
seed	integer, default: 42, set seed for generation of colors (n > given colors (palettes)) and colors ordering (see order)
order	default: 1, numeric (vector) to adjust colors order, -1: reverse order, 0: seeded random order, >1: shift order, c(-1, >1): reverse then shift order, or numeric vector as many colors to set custom order (if longer, vector shortened to n colors)
verbose	default: TRUE, mentions if and how many colors are generated

Value

sets colors in ggplot2 plotted object, see examples

See Also

Browse vignettes with `vignette("Introduction to coloRify")`

Examples

```
## viridis ggplot2 examples Colorified

## non-discrete
dat <- data.frame(x = rnorm(10000), y = rnorm(10000))

if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(dat, ggplot2::aes(x = x, y = y)) +
    ggplot2::geom_hex() + ggplot2::coord_fixed() +
    scale_fill_colorify(colors = 'viridis', n = 4) + ggplot2::theme_bw()

## discrete
```

```
  df <- data.frame(category = c("A", "B", "C", "D"), value = c(10, 23, 15, 8))


  ggplot2::ggplot(df, ggplot2::aes(x = category, y = value, fill = category)) +
    ggplot2::geom_bar(stat = "identity") +
    scale_fill_colorify(discrete = TRUE, colors = 'viridis')
}
```

Index