Package 'colorspace'

October 12, 2022

Version 2.0-3

Date 2022-02-19

Title A Toolbox for Manipulating and Assessing Colors and Palettes

Description Carries out mapping between assorted color spaces including RGB, HSV, HLS, CIEXYZ, CIELUV, HCL (polar CIELUV), CIELAB, and polar CIELAB. Qualitative, sequential, and diverging color palettes based on HCL colors are provided along with corresponding ggplot2 color scales. Color palette choice is aided by an interactive app (with either a Tcl/Tk or a shiny graphical user interface) and shiny apps with an HCL color picker and a color vision deficiency emulator. Plotting functions for displaying and assessing palettes include color swatches, visualizations of the HCL space, and trajectories in HCL and/or RGB spectrum. Color manipulation functions include: desaturation, lightening/darkening, mixing, and simulation of color vision deficiencies (deutanomaly, protanomaly, tritanomaly). Details can be found on the project web page at <https://colorspace.R-Forge.R-project.org/> and in the accompanying scientific paper: Zeileis et al. (2020, Journal of Statistical Software, <doi:10.18637/jss.v096.i01>).

Depends R (>= 3.0.0), methods

Imports graphics, grDevices, stats

Suggests datasets, utils, KernSmooth, MASS, kernlab, mvtnorm, vcd, tcltk, shiny, shinyjs, ggplot2, dplyr, scales, grid, png, jpeg, knitr, rmarkdown, RColorBrewer, rcartocolor, scico, viridis, wesanderson

VignetteBuilder knitr

License BSD_3_clause + file LICENSE

URL https://colorspace.R-Forge.R-project.org/, https://hclwizard.org/

BugReports https://colorspace.R-Forge.R-project.org/contact.html

LazyData yes

RoxygenNote 7.1.2

NeedsCompilation yes

Author Ross Ihaka [aut],
Paul Murrell [aut] (<https://orcid.org/0000-0002-3224-8858>),
Kurt Hornik [aut] (<https://orcid.org/0000-0003-4198-9911>),
Jason C. Fisher [aut] (<https://orcid.org/0000-0001-9032-8912>),
Reto Stauffer [aut] (<https://orcid.org/0000-0002-3798-5507>),
Claus O. Wilke [aut] (<https://orcid.org/0000-0002-7470-9261>),
Claire D. McWhite [aut] (<https://orcid.org/0000-0001-7346-3047>),
Achim Zeileis [aut, cre] (<https://orcid.org/0000-0003-0918-3766>)

Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>

Repository CRAN

Date/Publication 2022-02-21 09:50:02 UTC

R topics documented:

`adjust_transparency` *Adjust or Extract Transparency of Colors*

Description

Adjust (i.e., add, remove, or modify) or extract alpha transparency of a vector of colors.

Usage

```
adjust_transparency(col, alpha = TRUE)

extract_transparency(col, mode = "numeric", default = 1)
```

Arguments

`col`	vector of R colors. Can be any of the three kinds of R colors, i.e., either a color name (an element of `colors`), a hexadecimal (hex) string of the form `"#rrggbb"` or `"#rrggbbaa"` (see `rgb`), or an integer `i` meaning `palette()[i]`. Additionally, `col` can be a formal `color-class` object or a matrix with three rows containing R/G/B (0-255) values.
`alpha`	either a new alpha transparency value or logical (to add/remove alpha) or `NULL`. See details.
`mode`	character specifying the output mode for the alpha transparency, can be `"numeric"`, `"integer"`, `"character"` or `"hexmode"`. See details.
`default`	vector of length 1 specifying the default alpha transparency that should be returned for colors that do not specify any explicitly (defaulting to fully opaque). Can either be numeric, integer, character, or hexmode.

Details

Alpha transparency is useful for making colors semi-transparent, e.g., for overlaying different elements in graphics. An alpha value of 0 (or 00 in hex strings) corresponds to fully transparent and an alpha value of 1 (or FF in hex strings) corresponds to fully opaque. If a color hex string in R does not provide an explicit alpha transparency, the color is assumed to be fully opaque.

The `adjust_transparency` function can be used to adjust the alpha transparency of a set of colors. It always returns a hex color specification. This hex color can have the alpha transparency added/removed/modified depending on the specification of `alpha`:

- `alpha = NULL`: Returns a hex vector with alpha transparency only if needed. Thus, it keeps the alpha transparency for the colors (if any) but only if different from opaque.
- `alpha = TRUE`: Returns a hex vector with alpha transparency for all colors, using opaque (FF) as the default if missing.
- `alpha = FALSE`: Returns a hex vector without alpha transparency for all colors (even if the original colors had non-opaque alpha).
- `alpha` numeric: Returns a hex vector with alpha transparency for all colors set to the `alpha` argument (recycled if necessary).

The `extract_transparency` function can be used to extract the alpha transparency from a set of colors. It allows to specify the `default` value - that should be used for colors without an explicit alpha transparency (defaulting to fully opaque) - and `mode` of the return value. This can either be numeric (in [0, 1]), integer (0L, 1L, ..., 255L), character ("00", "01", ..., "FF"), or an object of class `hexmode` (internally represented as integer with printing as character). The `default` can use any of these modes as well (independent of the output `mode`) or be `NA`.

Value

For `adjust_transparency` character vector with hexadecimal color strings with alpha transparency corresponding to `alpha` argument. For `extract_transparency` a vector of alpha transparency values with the indicated `mode`.

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

`rgb`, `desaturate`, `lighten`

Examples

```
## modify transparency of a color (in different formats)
adjust_transparency("black",   alpha = c(0, 0.5, 1)) ## name
adjust_transparency("#000000", alpha = c(0, 0.5, 1)) ## hex string
adjust_transparency(1,         alpha = c(0, 0.5, 1)) ## palette() integer

## three shades of gray (in different formats:
```

```
## name/opaque, hex/opaque, hex/semi-transparent)
x <- c("gray", "#BEBEBE", "#BEBEBE80")

## adjust transparency
adjust_transparency(x, alpha = NULL)  ## only if necessary
adjust_transparency(x, alpha = TRUE)  ## add
adjust_transparency(x, alpha = FALSE) ## remove
adjust_transparency(x, alpha = 0.8)   ## modify

## extract transparency in different formats
extract_transparency(x, mode = "numeric") ## default
extract_transparency(x, mode = "integer")
extract_transparency(x, mode = "character")
extract_transparency(x, mode = "hexmode")

## extract transparency with different default values
extract_transparency(x, default = NA)
extract_transparency(x, default = 0.5)
extract_transparency(x, default = 128L)
extract_transparency(x, default = "80", mode = "integer")
```

`choose_palette` *Graphical User Interface for Choosing HCL Color Palettes*

Description

A graphical user interface (GUI) for viewing, manipulating, and choosing HCL color palettes.

Usage

```
choose_palette(pal = diverging_hcl, n = 7L, parent = NULL, gui = "tcltk", ...)

hclwizard(n = 7L, gui = "shiny", ...)
```

Arguments

`pal`	function; the initial palette, see 'Value' below. Only used if `gui = "tcltk"`.
`n`	integer; the initial number of colors in the palette.
`parent`	tkwin; the GUI parent window. Only used if `gui = "tcltk"`.
`gui`	character; GUI to use. Available options are `tcltk` and `shiny`, see 'Details' below.
`...`	used for development purposes only.

Details

Computes palettes based on the HCL (hue-chroma-luminance) color model (as implemented by `polarLUV`). The GUIs interface the palette functions `qualitative_hcl` for qualitative palettes, `sequential_hcl` for sequential palettes with a single or multiple hues, and `diverging_hcl` for diverging palettes (composed from two single-hue sequential palettes).

Two different GUIs are implemented and can be selected using the function input argument `gui` (`"tcltk"` or `"shiny"`). Both GUIs allows for interactive modification of the arguments of the respective palette-generating functions, i.e., starting/ending hue (wavelength, type of color), minimal/maximal chroma (colorfulness), minimal maximal luminance (brightness, amount of gray), and a power transformations that control how quickly/slowly chroma and/or luminance are changed through the palette. Subsets of the parameters may not be applicable depending on the type of palette chosen. See `qualitative_hcl` and Zeileis et al. (2009, 2019) for a more detailed explanation of the different arguments. Stauffer et al. (2015) provide more examples and guidance.

Optionally, active palette can be illustrated by using a range of examples such as a map, heatmap, scatter plot, perspective 3D surface etc.

To demonstrate different types of deficiencies, the active palette may be desaturated (emulating printing on a grayscale printer) and collapsed to emulate different types of color-blindness (without red-green or green-blue contrasts) using the `simulate_cvd` functions.

`choose_palette` by default starts the Tcl/Tk version of the GUI while `hclwizard` by default starts the shiny version. `hcl_wizard` is an alias for `hclwizard`.

Value

Returns a palette-generating function with the selected arguments. Thus, the returned function takes an integer argument and returns the corresponding number of HCL colors by traversing HCL space through interpolation of the specified hue/chroma/luminance/power values.

Author(s)

Jason C. Fisher, Reto Stauffer, Achim Zeileis

References

Zeileis A, Hornik K, Murrell P (2009). Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis*, **53**, 3259–3270. doi: 10.1016/j.csda.2008.11.033 Preprint available from https://www.zeileis.org/papers/Zeileis+Hornik+Murrell-2009.pdf.

Stauffer R, Mayr GJ, Dabernig M, Zeileis A (2015). Somewhere over the Rainbow: How to Make Effective Use of Colors in Meteorological Visualizations. *Bulletin of the American Meteorological Society*, **96**(2), 203–216. doi: 10.1175/BAMSD1300155.1

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

`simulate_cvd`, `desaturate`, `qualitative_hcl`.

Examples

```
if(interactive()) {
## Using tcltk GUI
pal <- choose_palette()
## or equivalently: hclwizard(gui = "tcltk")

## Using shiny GUI
pal <- hclwizard()
## or equivalently: choose_palette(gui = "shiny")

## use resulting palette function
filled.contour(volcano, color.palette = pal, asp = 1)
}
```

`color-class` *Class "color"*

Description

Objects from the class *color* represent colors in a number of color spaces. In particular, there are subclasses of color which correspond to RGB, HSV, HLS, CIEXYZ, CIELUV, CIELAB and polar versions of the last two spaces.

Objects from the Class

Objects can be created by calls to the functions `RGB`, `sRGB`, `HSV`, `HLS`, `XYZ`, `LUV`, `LAB`, `polarLUV`, and `polarLAB`. These are all subclasses of the virtual class *color*.

Slots

`coords`**:** An object of class `"matrix"`.

Methods

[`signature(x = "color")`: This method makes it possible to take subsets of a vector of colors.

coerce `signature(from = "color", to = "RGB")`: convert a color vector to RGB.

coerce `signature(from = "color", to = "sRGB")`: convert a color vector to sRGB.

coerce `signature(from = "color", to = "XYZ")`: convert a color vector to XYZ.

coerce `signature(from = "color", to = "LAB")`: convert a color vector to LAB.

coerce `signature(from = "color", to = "polarLAB")`: convert a color vector to polarLAB.

coerce `signature(from = "color", to = "HSV")`: convert a color vector to HSV.

coerce `signature(from = "color", to = "HLS")`: convert a color vector to HLS.

coerce `signature(from = "color", to = "LUV")`: convert a color vector to LUV.

coerce `signature(from = "color", to = "polarLUV")`: convert a color vector to polarLUV.

coords `signature(color = "color")`: extract the color coordinates from a color vector.

plot `signature(x = "color")`: plot a color vector

show `signature(object = "color")`: show a color vector.

Author(s)

Ross Ihaka

See Also

RGB, XYZ, HSV, HLS, LAB, polarLAB, LUV, polarLUV, mixcolor.

Examples

```
x <- sRGB(runif(1000), runif(1000), runif(1000))
plot(as(x, "LUV"))
```

`contrast_ratio` *W3C Contrast Ratio*

Description

Compute (and visualize) the contrast ratio of pairs of colors, as defined by the World Wide Web Consortium (W3C).

Usage

```
contrast_ratio(
  col,
  col2 = "white",
  algorithm = c("WCAG", "APCA"),
  plot = FALSE,
  border = FALSE,
  cex = 2,
  off = 0.05,
  mar = rep(0.5, 4),
  digits = 2L,
  ...
)
```

Arguments

`col, col2`	vectors of any of the three kind of R colors, i.e., either a color name (an element of colors), a hexadecimal string of the form `"#rrggbb"` or `"#rrggbbaa"` (see rgb), or an integer `i` meaning `palette()[i]`. Both can be vectors and are recycled as necessary.
`algorithm`	character specifying whether the established standard `"WCAG"` 2.1 algorithm should be used or the improved `"APCA"` 0.98G-4g algorithm, still under development.
`plot`	logical indicating whether the contrast ratios should also be visualized by simple color swatches. Can also be a vector of length 2, indicating whether the foreground color should be visualized on the background color and/or the background color on the foreground color.

border	logical or color specification for the borders around the color swatches (only used if plot = TRUE). The default is FALSE which is equivalent to "transparent". If TRUE the border is drawn in the same color as the text in the rectangle.
cex	numeric. Size of the text in the color color swatches (only if plot = TRUE).
off	numeric. Vertical offset between the different color swatches (only if plot = TRUE). Can also be of length 2 giving both vertical and horizontal offsets, respectively.
mar	numeric. Size of the margins around the color swatches (only if plot = TRUE).
digits	numeric. Number of digits for the contrast ratios displayed in the color swatches (only if plot = TRUE)
...	further arguments passed to the plot of the color swatches (only if plot = TRUE).

Details

The W3C Content Accessibility Guidelines (WCAG) recommend a contrast ratio of at least 4.5 for the color of regular text on the background color, and a ratio of at least 3 for large text. See https://www.w3.org/TR/WCAG21/#contrast-minimum.

The contrast ratio is defined in https://www.w3.org/TR/WCAG21/#dfn-contrast-ratio as `(L1 + 0.05) / (L2 + 0.05)` where `L1` and `L2` are the relative luminances (see https://www.w3.org/TR/WCAG21/#dfn-relative-luminance) of the lighter and darker colors, respectively. The relative luminances are weighted sums of scaled sRGB coordinates: `0.2126 * R + 0.7152 * G + 0.0722 * B` where each of `R`, `G`, and `B` is defined as `ifelse(RGB <= 0.03928, RGB/12.92, ((RGB + 0.055)/1.055)^2.4)` based on the `RGB` coordinates between 0 and 1.

For use in the next major revision of the WCAG a new advanced perceptual contrast algorithm (APCA) has been proposed by Somers (2022), see also Muth (2022) for more background and details. APCA is still under development, here version 0.98G-4g is implemented. Unlike the standard WCAG algorithm, APCA takes into account which color is the text and which is the background. Hence for the APCA algorithm a matrix with normal and reverse polarity is returned. An absolute value of 45 is "sort of" like a WCAG ratio of 3, 60 is "sort of" like 4.5.

Value

A numeric vector with the contrast ratios is returned (invisibly, if `plot` is `TRUE`).

References

W3C (2018). "Web Content Accessibility Guidelines (WCAG) 2.1." https://www.w3.org/TR/WCAG21/

Somers A (2022). "Advanced Perceptual Contrast Algorithm." https://github.com/Myndex/SAPC-APCA

Muth LC (2022). "It's Time for a More Sophisticated Color Contrast Check for Data Visualizations." Datawrapper Blog. https://blog.datawrapper.de/color-contrast-check-data-vis-wcag-apca/

See Also

desaturate

Examples

```
# check contrast ratio of default palette on white background
contrast_ratio(palette(), "white")

# visualize contrast ratio of default palette on white and black background
contrast_ratio(palette(), "white", plot = TRUE)
contrast_ratio(palette()[-1], "black", plot = TRUE)

# APCA algorithm
contrast_ratio(palette(), "white", algorithm = "APCA")
contrast_ratio(palette(), "white", algorithm = "APCA", plot = TRUE, digits = 0)
```

---

coords *Extract the Numerical Coordinates of a Color*

Description

This function returns a matrix with three columns which give the coordinates of a color in its natural color space.

Usage

```
coords(color)
```

Arguments

color	A color.

Value

A numeric matrix giving the coordinates of the color.

Author(s)

Ross Ihaka

See Also

RGB, XYZ, LAB, polarLAB, LUV, polarLUV, mixcolor.

Examples

```
x <- sRGB(1, 0, 0)
coords(as(x, "HSV"))
```

cvd *Color Vision Deficiency (CVD) Conversion Tables*

Description

Conversion tables for simulating different types of color vision deficiency (CVD): Protanomaly, deutanomaly, tritanomaly.

Usage

```
protanomaly_cvd

deutanomaly_cvd

tritanomaly_cvd
```

Format

Lists of 3x3 RGB-color transformation matrices for the various types of CVD. Each list contains 11 transformation matrices representing increasingly severe color vision deficiency.

Details

Machado et al. (2009) have established a novel model, that allows to handle normal color vision, anomalous trichromacy, and dichromacy in a unified way. They also provide conversion formulas along with tables of certain constants that allow to simulate various types of CVD. See `simulate_cvd` for the corresponding simulation functions.

References

Machado GM, Oliveira MM, Fernandes LAF (2009). A Physiologically-Based Model for Simulation of Color Vision Deficiency. *IEEE Transactions on Visualization and Computer Graphics*. **15**(6), 1291–1298. doi: 10.1109/TVCG.2009.113 Online version with supplements at http://www.inf.ufrgs.br/~oliveira/pubs_files/CVD_Simulation/CVD_Simulation.html.

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

`simulate_cvd`

cvd_emulator *Graphical User Interface to Check Images for Color Constraints*

Description

A graphical user interface (GUI) to check an existing jpg/png image for (possible) color constraints. The image will be converted to protanope vision, deuteranope vision, and a desaturated version (monochromatic vision). Allows a rapid check whether the colors used in the image show some constraints with respect to color deficiency or color blindness.

Usage

```
cvd_emulator(file, overwrite = FALSE, shiny.trace = FALSE)
```

Arguments

`file`	If not set, an interactive GUI will be started. If `x` is of type `character` it has to be the full path to an image of type png or jpg/jpeg. The image will be converted and stored on disc, no GUI.
`overwrite`	`logical`. Only used if `file` is provided. Allow the function to overwrite files on disc if they exist.
`shiny.trace`	`logical`. Can be set to `TRUE` for more verbose output when the GUI is started (development flag).

Author(s)

Reto Stauffer, Claus O. Wilke, Achim Zeileis

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

cvd_image *Convert Colors of an Image*

Description

Used in `cvd_emulator`. Takes an image object and converts the colors using `deutan`, `protan`, `tritan`, `desaturate` functions. The image will be written to disc as a PNG file.

Usage

```
cvd_image(img, type, file, severity = 1)
```

Arguments

img	array as returned by readPNG and readJPEG of size height x width x depth. The depth coordinate contains R/G/B and alpha if given (png).
type	string name of the function which will be used to convert the colors (deutan, protan, tritan, desaturate). If set to original the image will be written as is.
file	string with (full) path to resulting image. Has to be a png image name.
severity	numeric. Severity of the color vision defect, a number between 0 and 1.

---

demoplot *Color Palette Demonstration Plot*

Description

Demonstration of color palettes in various kinds of statistical graphics.

Usage

```
demoplot(
  x,
 type = c("map", "heatmap", "scatter", "spine", "bar", "pie", "perspective", "mosaic",
    "lines"),
  ...
)
```

Arguments

x	character vector containing color hex codes.
type	character indicating the type of demonstration plot.
...	currently not used.

Details

To demonstrate how different kinds of color palettes work in different kinds of statistical displays, demoplot provides a simple convenience interface to some base graphics with (mostly artificial) data sets. All types of demos can deal with arbitrarily many colors. However, some displays are much more suitable for a low number of colors (e.g., the pie chart) while others work better with more colors (e.g., the heatmap).

Value

demoplot returns invisibly what the respective base graphics functions return that are called internally.

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

specplot, hclplot

Examples

```
## all built-in demos with the same sequential heat color palette
par(mfrow = c(3, 3))
cl <- sequential_hcl(5, "Heat")
for (i in c("map", "heatmap", "scatter", "spine", "bar", "pie", "perspective", "mosaic", "lines")) {
  demoplot(cl, type = i)
}

## qualitative palettes: light pastel colors for shading areas (pie)
## and darker colorful palettes for points or lines
demoplot(qualitative_hcl(4, "Pastel 1"), type = "pie")
demoplot(qualitative_hcl(4, "Set 2"), type = "scatter")
demoplot(qualitative_hcl(4, "Dark 3"), type = "lines")

## sequential palettes: display almost continuous gradients with
## strong luminance contrasts (heatmap, perspective) and colorful
## sequential palette for spine plot with only a few ordered categories
demoplot(sequential_hcl(99, "Purple-Blue"), type = "heatmap")
demoplot(sequential_hcl(99, "Reds"), type = "perspective")
demoplot(sequential_hcl(4, "Viridis"), type = "spine")

## diverging palettes: display almost continuous gradient with
## strong luminance contrast bringing out the extremes (map),
## more colorful palette with lower luminance contrasts for displays
## with fewer colors (mosaic, bar)
demoplot(diverging_hcl(99, "Tropic", power = 2.5), type = "map")
demoplot(diverging_hcl(5, "Green-Orange"), type = "mosaic")
demoplot(diverging_hcl(5, "Blue-Red 2"), type = "bar")

## some palettes that work well on black backgrounds
par(mfrow = c(2, 3), bg = "black")
demoplot(sequential_hcl(9, "Oslo"), "heatmap")
demoplot(sequential_hcl(9, "Turku"), "heatmap")
demoplot(sequential_hcl(9, "Inferno", rev = TRUE), "heatmap")
demoplot(qualitative_hcl(9, "Set 2"), "lines")
demoplot(diverging_hcl(9, "Berlin"), "scatter")
demoplot(diverging_hcl(9, "Cyan-Magenta", l2 = 20), "lines")
```

`desaturate` *Desaturate Colors by Chroma Removal in HCL Space*

Description

Transform a vector of given colors to the corresponding colors with chroma reduced (by a tunable amount) in HCL space.

Usage

```
desaturate(col, amount = 1, ...)
```

Arguments

`col`	vector of R colors. Can be any of the three kinds of R colors, i.e., either a color name (an element of `colors`), a hexadecimal string of the form `"#rrggbb"` or `"#rrggbbaa"` (see `rgb`), or an integer `i` meaning `palette()[i]`. Input `col` can also be a matrix with three rows containing R/G/B (0-255) values, see details.
`amount`	numeric specifying the amount of desaturation where `1` corresponds to complete desaturation, `0` to no desaturation, and values in between to partial desaturation.
`...`	additional arguments. If `severity` is specified it will overrule the input argument `amount` (for convenience).

Details

If input `col` is a vector given colors are first transformed to RGB (either using `hex2RGB` or `col2rgb`) and then to HCL (`polarLUV`). In HCL, chroma is reduced and then the color is transformed back to a hexadecimal string.

If input `col` is a matrix with three rows named `R`, `G`, and `B` (top down) they are interpreted as Red-Green-Blue values within the range `[0-255]`. The desaturation takes place in the HCL space as well. Instead of an (s)RGB color vector a matrix of the same size as the input `col` with desaturated Red-Green-Blue values will be returned. This can be handy to avoid too many conversions.

Value

A character vector with (s)RGB codings of the colors in the palette if input `col` is a vector. If input `col` is a matrix with R/G/B values a matrix of the same form and size will be returned.

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

`polarLUV`, `hex`, `lighten`

Examples

```
## rainbow of colors and their desaturated counterparts
rainbow_hcl(12)
desaturate(rainbow_hcl(12))

## convenience demo function
wheel <- function(col, radius = 1, ...)
  pie(rep(1, length(col)), col = col, radius = radius, ...)

## compare base and colorspace palettes
## (in color and desaturated)
par(mar = rep(0, 4), mfrow = c(2, 2))
## rainbow color wheel
wheel(rainbow_hcl(12))
wheel(rainbow(12))
wheel(desaturate(rainbow_hcl(12)))
wheel(desaturate(rainbow(12)))

## apply desaturation directly on RGB values
RGB <- t(hex2RGB(rainbow(3))@coords * 255)
desaturate(RGB)
```

`divergingx_hcl` *(More) Flexible Diverging HCL Palettes*

Description

Diverging HCL color palettes generated through combination of two fully flexible (and possibly unbalanced) multi-hue sequential palettes.

Usage

```
divergingx_hcl(
  n,
  palette = "Geyser",
  ...,
  fixup = TRUE,
  alpha = 1,
  rev = FALSE,
  h1,
  h2,
  h3,
  c1,
  c2,
  c3,
  l1,
  l2,
  l3,
```

```
  p1,
  p2,
  p3,
  p4,
  cmax1,
  cmax2
)

divergingx_palettes(palette = NULL, plot = FALSE, n = 7L, ...)
```

Arguments

n	the number of colors (≥ 1) to be in the palette.
palette	character with the name (see details).
...	arguments passed to hex.
fixup	logical. Should the color be corrected to a valid RGB value?
alpha	numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque).
rev	logical. Should the palette be reversed?
h1	numeric. Starting hue coordinate.
h2	numeric. Center hue coordinate.
h3	numeric. Ending hue coordinate.
c1	numeric. Chroma coordinate corresponding to h1.
c2	numeric. Chroma coordinate corresponding to h2 (if NA, set to 0).
c3	numeric. Chroma coordinate corresponding to h3.
l1	numeric. Luminance coordinate corresponding to h1.
l2	numeric. Luminance coordinate corresponding to h2.
l3	numeric. Luminance coordinate corresponding to h3 (if NA, l1 is used).
p1	numeric. Power parameter for chroma coordinates in first sequential palette.
p2	numeric. Power parameter for luminance coordinates in first sequential palette (if NA, p1 is used).
p3	numeric. Power parameter for chroma coordinates in second sequential palette (if NA, p1 is used).
p4	numeric. Power parameter for luminance coordinates in second sequential palette (if NA, p3 is used).
cmax1	numeric. Maximum chroma coordinate in first sequential palette (not used if NA).
cmax2	numeric. Maximum chroma coordinate in second sequential palette (if NA, cmax1 is used).
plot	logical. Should the selected HCL color palettes be visualized?

Details

The `divergingx_hcl` function simply calls `sequential_hcl` twice with a prespecified set of hue, chroma, and luminance parameters. This is similar to `diverging_hcl` but allows for more flexibility: `diverging_hcl` employs two *single-hue* sequential palettes, always uses zero chroma for the neutral/central color, and restricts the chroma/luminance path to be the same in both "arms" of the palette. In contrast, `divergingx_hcl` relaxes this to two full *multi-hue* palettes that can thus go through a non-gray neutral color (typically light yellow). Consequently, the chroma/luminance paths can be rather unbalanced between the two arms.

With this additional flexibility various diverging palettes suggested by https://ColorBrewer2.org/ and CARTO (https://carto.com/carto-colors/), can be emulated along with the Zissou 1 palette from **wesanderson**, Cividis from **viridis**, and Roma from **scico**.

Available CARTO palettes: ArmyRose, Earth, Fall, Geyser, TealRose, Temps, and Tropic (with Tropic also available in `diverging_hcl`).

Available ColorBrewer.org palettes: PuOr, RdBu, RdGy, PiYG, PRGn, BrBG, RdYlBu, RdYlGn, Spectral.

Value

A character vector with (s)RGB codings of the colors in the palette.

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

`sequential_hcl`, `diverging_hcl`

Examples

```
## show emulated CARTO/ColorBrewer.org palettes
divergingx_palettes(plot = TRUE)

## compared to diverging_hcl() the diverging CARTO palettes are typically warmer
## but also less balanced with respect to chroma/luminance, see e.g.,
specplot(divergingx_hcl(7, "ArmyRose"))
```

`hclplot` *Palette Plot in HCL Space*

Description

Visualization of color palettes in HCL space projections.

Usage

```
hclplot(
  x,
  type = NULL,
  h = NULL,
  c = NULL,
  l = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  cex = 1,
  axes = TRUE,
  bg = "white",
  lwd = 1,
  size = 2.5,
  ...
)
```

Arguments

x	character vector containing color hex codes, or a color-class object.
type	type character specifying which type of palette should be visualized ("qualitative", "sequential", or "diverging"). For qualitative palettes a hue-chroma plane is used, otherwise a chroma-luminance plane. By default, the type is inferred from the luminance trajectory corresponding to x.
h	numeric hue(s) to be used for type = "sequential" and type = "diverging". By default, these are inferred from the colors in x.
c	numeric. Maximal chroma value to be used.
l	numeric luminance(s) to be used for type = "qualitative". By default, this is inferred from the colors in x.
xlab, ylab, main	character strings for annotation, by default generated from the type of color palette visualized.
cex	numeric character extension.
axes	logical. Should axes be drawn?
bg, lwd, size	graphical control parameters for the color palette trajectory.
...	currently not used.

Details

The function hclplot is an auxiliary function for illustrating the trajectories of color palettes in two-dimensional HCL space projections. It collapses over one of the three coordinates (either the hue H or the luminance L) and displays a heatmap of colors combining the remaining two dimensions. The coordinates for the given color palette are highlighted to bring out its trajectory.

The function hclplot has been designed to work well with the hcl_palettes in this package. While it is possible to apply it to other color palettes as well, the results might look weird or

confusing if these palettes are constructed very differently (e.g., as in the highly saturated base R palettes).

More specifically, the following palettes can be visualized well:

- Qualitative with (approximately) constant luminance. In this case, `hclplot` shows a hue-chroma plane (in polar coordinates), keeping luminance at a fixed level (by default displayed in the main title of the plot). If the luminance is, in fact, not approximately constant, the luminance varies along with hue and chroma, using a simple linear function (fitted by least squares). `hclplot` shows a chroma-luminance plane, keeping hue at a fixed level (by default displayed in the main title of the plot). If the hue is, in fact, not approximately constant, the hue varies along with chroma and luminance, using a simple linear function (fitted by least squares.
- Diverging with two (approximately) constant hues: This case is visualized with two back-to-back sequential displays.

To infer the type of display to use, by default, the following heuristic is used: If luminance is not approximately constant (range > 10) and follows rougly a triangular pattern, a diverging display is used. If luminance is not constant and follows roughly a linear pattern, a sequential display is used. Otherwise a qualitative display is used.

Value

`hclplot` invisibly returns a matrix with the HCL coordinates corresponding to `x`.

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

`specplot`

Examples

```
## for qualitative palettes luminance and chroma are fixed, varying only hue
hclplot(qualitative_hcl(9, c = 50, l = 70))

## single-hue sequential palette (h = 260) with linear vs. power-transformed trajectory
hclplot(sequential_hcl(7, h = 260, c = 80, l = c(35, 95), power = 1))
hclplot(sequential_hcl(7, h = 260, c = 80, l = c(35, 95), power = 1.5))

## advanced single-hue sequential palette with triangular chroma trajectory
## (piecewise linear vs. power-transformed)
hclplot(sequential_hcl(7, h = 245, c = c(40, 75, 0), l = c(30, 95), power = 1))
hclplot(sequential_hcl(7, h = 245, c = c(40, 75, 0), l = c(30, 95), power = c(0.8, 1.4)))

## multi-hue sequential palette with small hue range and triangular chroma vs.
## large hue range and linear chroma trajectory
hclplot(sequential_hcl(7, h = c(260, 220), c = c(50, 75, 0), l = c(30, 95), power = 1))
```

```
hclplot(sequential_hcl(7, h = c(260, 60), c = 60, l = c(40, 95), power = 1))

## balanced diverging palette constructed from two simple single-hue sequential
## palettes (for hues 260/blue and 0/red)
hclplot(diverging_hcl(7, h = c(260, 0), c = 80, l = c(35, 95), power = 1))
```

`hcl_color_picker` *Graphical User Interface to Pick Colors in HCL Space*

Description

The app visualizes colors either along the hue-chroma plane for a given luminance value or along the luminance-chroma plane for a given hue. Colors can be entered by specifying the hue (H), chroma (C), and luminance (L) values via sliders, by entering an RGB hex code, or by clicking on a color in the hue-chroma or luminance-chroma plane. It is also possible to select individual colors and add them to a palette for comparison and future reference.

Usage

```
hcl_color_picker(shiny.trace = FALSE)

choose_color(shiny.trace = FALSE)
```

Arguments

`shiny.trace`	logical: used for debugging the shiny interface.

Details

`choose_color` is a convenience alias for `hcl_color_picker` to go along with `choose_palette`. Another alias is `hclcolorpicker`.

Value

`hclcolorpicker` invisibly returns a vector of colors choosen. If no colors have been selected `NULL` will be returned.

Author(s)

Claus O. Wilke, Reto Stauffer, Achim Zeileis

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

choose_palette

Examples

```
## Not run:
hcl_color_picker()

## End(Not run)
```

---

hcl_palettes *HCL Color Palettes*

Description

Qualitative, sequential (single-hue and multi-hue), and diverging color palettes based on the HCL (hue-chroma-luminance) color model.

Usage

```
hcl_palettes(type = NULL, palette = NULL, plot = FALSE, n = 5L, ...)

## S3 method for class 'hcl_palettes'
print(x, ...)

## S3 method for class 'hcl_palettes'
summary(object, ...)

## S3 method for class 'hcl_palettes'
plot(x, n = 5L, fixup = TRUE, off = NULL, border = NULL, ...)

qualitative_hcl(
  n,
  h = c(0, 360 * (n - 1)/n),
  c = 80,
  l = 60,
  fixup = TRUE,
  alpha = 1,
  palette = NULL,
  rev = FALSE,
  register = "",
  ...,
  h1,
  h2,
  c1,
  l1
)
```

```
sequential_hcl(
  n,
  h = 260,
  c = 80,
  l = c(30, 90),
  power = 1.5,
  gamma = NULL,
  fixup = TRUE,
  alpha = 1,
  palette = NULL,
  rev = FALSE,
  register = "",
  ...,
  h1,
  h2,
  c1,
  c2,
  l1,
  l2,
  p1,
  p2,
  cmax,
  c.
)

diverging_hcl(
  n,
  h = c(260, 0),
  c = 80,
  l = c(30, 90),
  power = 1.5,
  gamma = NULL,
  fixup = TRUE,
  alpha = 1,
  palette = NULL,
  rev = FALSE,
  register = "",
  ...,
  h1,
  h2,
  c1,
  l1,
  l2,
  p1,
  p2,
  cmax
)
```

Arguments

type	character indicating type of HCL palette.
palette	character. Name of HCL color palette.
plot	logical. Should the selected HCL color palettes be visualized?
n	the number of colors (≥ 1) to be in the palette.
...	Other arguments passed to hex.
x, object	A hcl_palettes object.
fixup	logical. Should the color be corrected to a valid RGB value?
off	numeric. Vector of length 2 indicating horizontal and vertical offsets between the color rectangles displayed.
border	character. Color of rectangle borders.
h, h1, h2	hue value in the HCL color description, has to be in [0, 360].
c, c., c1, c2	chroma value in the HCL color description.
l, l1, l2	luminance value in the HCL color description.
alpha	numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque).
rev	logical. Should the color palette vector be returned in reverse order?
register	character. If set to a non-empty character string, the corresponding palette is registered with that name for subsequent use (within the same session).
power, p1, p2	control parameter determining how chroma and luminance should be increased (1 = linear, 2 = quadratic, etc.).
gamma	Deprecated.
cmax	Maximum chroma value in the HCL color description.

Details

The HCL (hue-chroma-luminance) color model is a perceptual color model obtained by using polar coordinates in CIE LUV space (i.e., polarLUV), where steps of equal size correspond to approximately equal perceptual changes in color. By taking polar coordinates the resulting three dimensions capture the three perceptual axes very well: hue is the type of color, chroma the colorfulness compared to the corresponding gray, and luminance the brightness. This makes it relatively easy to create balanced palettes through trajectories in this HCL space. In contrast, in the more commonly-used HSV (hue-saturation-value) model (a simple transformation of RGB), the three axes are confounded so that luminance changes along with the hue leading to very unbalanced palettes (see rainbow_hcl for further illustrations).

Three types of palettes are derived based on the HCL model:

- Qualitative: Designed for coding categorical information, i.e., where no particular ordering of categories is available and every color should receive the same perceptual weight.
- Sequential: Designed for coding ordered/numeric information, i.e., where colors go from high to low (or vice versa).
- Diverging: Designed for coding numeric information around a central neutral value, i.e., where colors diverge from neutral to two extremes.

The corresponding functions are `qualitative_hcl`, `sequential_hcl`, and `diverging_hcl`. Their construction principles are explained in more detail below. At the core is the luminance axis (i.e., light-dark contrasts): These are easily decoded by humans and matched to high-low differences in the underlying data. Therefore, `sequential_hcl` palettes are always based on a *monotonic* luminance sequence whereas the colors in a `qualitative_hcl` palette all have the *same* luminance. Finally, `diverging_hcl` palettes use the same monotonic luminance sequence in both "arms" of the palette, i.e., correspond to two balanced sequential palettes diverging from the same neutral value. The other two axes, hue and chroma, are used to enhance the luminance information and/or to further discriminate the color.

All three palette functions specify trajectories in HCL space and hence need either single values or intervals of the coordinates `h`, `c`, `l`. Their interfaces are always designed such that `h`, `c`, `l` can take vector arguments (as needed) but alternatively or additionally `h1`/`h2`, `c1`/`c2`/`cmax`, and `l1`/`l2` can be specified. If so, the latter coordinates overwrite the former.

`qualitative_hcl` distinguishes the underlying categories by a sequence of hues while keeping both chroma and luminance constant to give each color in the resulting palette the same perceptual weight. Thus, `h` should be a pair of hues (or equivalently `h1` and `h2` can be used) with the starting and ending hue of the palette. Then, an equidistant sequence between these hues is employed, by default spanning the full color wheel (i.e, the full 360 degrees). Chroma `c` (or equivalently `c1`) and luminance `l` (or equivalently `l1`) are constants.

`sequential_hcl` codes the underlying numeric values by a monotonic sequence of increasing (or decreasing) luminance. Thus, the `l` argument should provide a vector of length 2 with starting and ending luminance (equivalently, `l1` and `l2` can be used). Without chroma (i.e., `c = 0`), this simply corresponds to a grayscale palette like `gray.colors`. For adding chroma, a simple strategy would be to pick a single hue (via `h` or `h1`) and then decrease chroma from some value (`c` or `c1`) to zero (i.e., gray) along with increasing luminance. For bringing out the extremes (a dark high-chroma color vs. a light gray) this is already very effective. For distinguishing also colors in the middle two strategies can be employed: (a) Hue can be varied as well by specifying an interval of hues in `h` (or beginning hue `h1` and ending hue `h2`). (b) Instead of a decreasing chroma a triangular chroma trajectory can be employed from `c1` over `cmax` to `c2` (or equivalently a vector `c` of length 3). This yields high-chroma colors in the middle of the palette that are more easily distinguished from the dark and light extremes. Finally, instead of employing linear trajectories, power transformations are supported in chroma and luminance via a vector `power` (or separate `p1` and `p2`). If `power[2]` (or `p2`) for the luminance trajectory is missing, it defaults to `power[1]`/`p1` from the chroma trajectory.

`diverging_hcl` codes the underlying numeric values by a triangular luminance sequence with different hues in the left and in the right arm of the palette. Thus, it can be seen as a combination of two sequential palettes with some restrictions: (a) a single hue is used for each arm of the palette, (b) chroma and luminance trajectory are balanced between the two arms, (c) the neutral central value has zero chroma. To specify such a palette a vector of two hues `h` (or equivalently `h1` and `h2`), either a single chroma value `c` (or `c1`) or a vector of two chroma values `c` (or `c1` and `cmax`), a vector of two luminances `l` (or `l1` and `l2`), and power parameter(s) `power` (or `p1` and `p2`) are used. For more flexible diverging palettes without the restrictrictions above (and consequently more parameters) `divergingx_hcl` is available. For backward compatibility, `diverge_hcl` is a copy of `diverging_hcl`.

To facilitate using HCL-based palettes a wide range of example palettes are provided in the package and can be specified by a name instead of a set of parameters/coordinates. The examples have been taken from the literature and many approximate color palettes from other software packages such as ColorBrewer.org (**RColorBrewer**), CARTO colors (**rcartocolor**), **scico**, or **viridis**. The

function hcl_palettes can be used to query the available pre-specified palettes. It comes with a print method (listing names and types), a summary method (additionally listing the underlying parameters/coordinates), and a plot method that creates a swatchplot with suitable labels.

Value

qualitative_hcl, sequential_hcl, diverging_hcl return a vector of n color strings (hex codes).

The function hcl_palettes returns a data frame of class "hcl_palettes" where each row contains information about one of the requested palettes (name, type, HCL trajectory coordinates). Suitable print, summary, and plot methods are available.

References

Zeileis A, Hornik K, Murrell P (2009). Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis*, **53**, 3259–3270. doi: 10.1016/j.csda.2008.11.033 Preprint available from https://www.zeileis.org/papers/Zeileis+Hornik+Murrell-2009.pdf.

Stauffer R, Mayr GJ, Dabernig M, Zeileis A (2015). Somewhere Over the Rainbow: How to Make Effective Use of Colors in Meteorological Visualizations. *Bulletin of the American Meteorological Society*, **96**(2), 203–216. doi: 10.1175/BAMSD1300155.1

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

divergingx_hcl

Examples

```
## overview of all _named_ HCL palettes
hcl_palettes()

## visualize
hcl_palettes("qualitative", plot = TRUE)
hcl_palettes("sequential (single-hue)", n = 7, plot = TRUE)
hcl_palettes("sequential (multi-hue)", n = 7, plot = TRUE)
hcl_palettes("diverging", n = 7, plot = TRUE)

## inspect a specific palette
## (upper-case, spaces, etc. are ignored for matching)
hcl_palettes(palette = "Dark 2")
hcl_palettes(palette = "dark2")

## set up actual colors
qualitative_hcl(4, h = c(0, 288), c = 50, l = 60) ## by hand
qualitative_hcl(4, palette = "dark2")             ## by name
qualitative_hcl(4, palette = "dark2", c = 80)     ## by name plus modification

## how HCL palettes are constructed:
```

```
## by varying the perceptual properties via hue/chroma/luminance
swatchplot(
  "Hue"       = sequential_hcl(5, h = c(0, 300), c = c(60, 60), l = 65),
  "Chroma"    = sequential_hcl(5, h = 0, c = c(100, 0), l = 65, rev = TRUE, power = 1),
 "Luminance" = sequential_hcl(5, h = 260, c = c(25, 25), l = c(25, 90), rev = TRUE, power = 1),
  off = 0
)

## for qualitative palettes luminance and chroma are fixed, varying only hue
hclplot(qualitative_hcl(9, c = 50, l = 70))

## single-hue sequential palette (h = 260) with linear vs. power-transformed trajectory
hclplot(sequential_hcl(7, h = 260, c = 80, l = c(35, 95), power = 1))
hclplot(sequential_hcl(7, h = 260, c = 80, l = c(35, 95), power = 1.5))

## advanced single-hue sequential palette with triangular chroma trajectory
## (piecewise linear vs. power-transformed)
hclplot(sequential_hcl(7, h = 245, c = c(40, 75, 0), l = c(30, 95), power = 1))
hclplot(sequential_hcl(7, h = 245, c = c(40, 75, 0), l = c(30, 95), power = c(0.8, 1.4)))

## multi-hue sequential palette with small hue range and triangular chroma vs.
## large hue range and linear chroma trajectory
hclplot(sequential_hcl(7, h = c(260, 220), c = c(50, 75, 0), l = c(30, 95), power = 1))
hclplot(sequential_hcl(7, h = c(260, 60), c = 60, l = c(40, 95), power = 1))

## balanced diverging palette constructed from two simple single-hue sequential
## palettes (for hues 260/blue and 0/red)
hclplot(diverging_hcl(7, h = c(260, 0), c = 80, l = c(35, 95), power = 1))

## to register a particular adapted palette for re-use in the same session
## with a new name the register=... argument can be used once, e.g.,
diverging_hcl(7, palette = "Tropic", h2 = 0, register = "mytropic")

## subsequently palete="mytropic" is available in diverging_hcl() and the diverging
## ggplot2 scales such as scale_color_continuous_diverging() etc.
demoplot(diverging_hcl(11, "mytropic"), type = "map")

## to register this palette in all R sessions you could place the following
## code in a startup script (e.g., .Rprofile):
## colorspace::diverging_hcl(7, palette = "Tropic", h2 = 0, register = "mytropic")
```

---

hex *Convert Colors to Hexadecimal Strings*

Description

This functions converts color-class objects into hexadecimal strings.

Usage

```
hex(from, gamma = NULL, fixup = FALSE)
```

Arguments

from	The color object to be converted.
gamma	Deprecated.
fixup	Should the color be corrected to a valid RGB value before correction. The default is to convert out-of-gamut colors to the string "NA".

Details

The color objects are first converted to sRGB color objects. They are then multiplied by 255 and rounded to obtain an integer value. These values are then converted to hexadecimal strings of the form "#RRGGBB" and suitable for use as color descriptions for R graphics. Out of gamut values are either corrected to valid RGB values by translating the the individual primary values so that they lie between 0 and 255.

Value

A vector of character strings.

Author(s)

Ross Ihaka

See Also

hex2RGB, RGB, sRGB, HSV, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
hsv <- HSV(seq(0, 360, length.out = 7)[-7], 1, 1)
hsv
hex(hsv)
barplot(rep(1,6), col = hex(hsv))
```

---

hex2RGB *Convert Hexadecimal Color Specifications to sRGB Objects*

Description

This function takes a vector of strings of the form "#RRGGBB" (hexadecimal color descriptions) into sRGB objects.

Usage

```
hex2RGB(x, gamma = FALSE)
```

Arguments

x	a vector of hexadecimal color descriptions.
gamma	Whether to apply gamma-correction.

Details

This function converts device-dependent color descriptions of the form "#RRGGBB" into sRGB color descriptions (linearized if gamma is TRUE). The alpha channel will be ignored if given ("#RRGGBBAA").

Value

An sRGB object describing the colors.

Author(s)

Ross Ihaka

See Also

hex, RGB, sRGB, HSV, XYZ, polarLAB, LUV, polarLUV.

Examples

```
hex2RGB(c("#FF0000","#00FF00", "#0000FF50"))
```

HLS *Create HLS Colors*

Description

This function creates colors of class HLS; a subclass of the virtual color-class class.

Usage

```
HLS(H, L, S, names)
```

Arguments

H, L, S	These arguments give the hue, lightness, and saturation of the colors. The values can be provided in separate H, L and S vectors or in a three-column matrix passed as H.
names	A vector of names for the colors (by default the row names of H are used).

Details

This function creates colors in an HLS color space. The hues should lie between between 0 and 360, and the lightness and saturations should lie between 0 and 1.

HLS is a relative color space; it is a transformation of an RGB color space. Conversion of HLS colors to any other color space must first involve a conversion to a specific RGB color space, for example the standard sRGB color space (IEC standard 61966).

Value

An object of class HLS which inherits from class color.

Author(s)

Ross Ihaka

See Also

sRGB, RGB, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
# A rainbow of full-intensity hues
HLS(seq(0, 360, length.out = 13)[-13], 0.5, 1)
```

HSV *Create HSV Colors*

Description

This function creates colors of class HSV; a subclass of the virtual color-class class.

Usage

```
HSV(H, S, V, names)
```

Arguments

H, S, V	These arguments give the hue, saturation and value of the colors. The values can be provided in separate H, S and V vectors or in a three-column matrix passed as H.
names	A vector of names for the colors (by default the row names of H are used).

Details

This function creates colors in an HSV color space. The hues should lie between between 0 and 360, and the saturations and values should lie between 0 and 1.

HSV is a relative color space; it is a transformation of an RGB color space. Conversion of HSV colors to any other color space must first involve a conversion to a specific RGB color space, for example the standard sRGB color space (IEC standard 61966).

Value

An object of class HSV which inherits from class color.

Author(s)

Ross Ihaka

See Also

sRGB, RGB, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
# A rainbow of full-intensity hues
HSV(seq(0, 360, length.out = 13)[-13], 1, 1)
```

LAB *Create LAB Colors*

Description

This function creates colors of class "LAB"; a subclass of the virtual color-class class.

Usage

```
LAB(L, A, B, names)
```

Arguments

L, A, B	these arguments give the L, A and B coordinates of the colors. The values can be provided in separate L, A and B vectors or in a three-column matrix passed as L.
names	a vector of names for the colors (by default the row names of L are used).

Details

The L, A and B values give the coordinates of the colors in the CIE $L^*a^*b^*$ space. This is a transformation of the 1931 CIE XYZ space which attempts to produce perceptually based axes. Luminance takes values between 0 and 100, and the other coordinates typically take values between -100 and 100, although these values can also be exceeded by highly saturated colors. The a and b coordinates measure positions on green/red and blue/yellow axes.

Value

An object of class LAB which inherits from class color.

Author(s)

Ross Ihaka

See Also

RGB, HSV, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
## Show the LAB space
set.seed(1)
x <- sRGB(runif(1000), runif(1000), runif(1000))
y <- as(x, "LAB")
head(x)
head(y)
plot(y)
```

---

lighten *Algorithmically Lighten or Darken Colors*

Description

The functions lighten and darken take a vector of R colors and adjust the colors such that they appear lightened or darkened, respectively.

Usage

```
lighten(
  col,
  amount = 0.1,
  method = c("relative", "absolute"),
  space = c("HCL", "HLS", "combined"),
  fixup = TRUE
)

darken(col, amount = 0.1, space = "combined", ...)
```

Arguments

col	vector of any of the three kind of R colors, i.e., either a color name (an element of colors), a hexadecimal string of the form "#rrggbb" or "#rrggbbaa" (see rgb), or an integer i meaning palette()[i].
amount	numeric specifying the amount of lightening. This is applied either multiplicatively or additively to the luminance value, depending on the setting of method (either relative or absolute). Negative numbers cause darkening.
method	character string specifying the adjustment method. Can be either "relative" or "absolute".
space	character string specifying the color space in which adjustment happens. Can be either "HLS" or "HCL".

fixup	logical If set to TRUE, colors that fall outside of the RGB color gamut are slightly modified by translating individual primary values so they lie between 0 and 255. If set to FALSE, out-of-gamut colors are replaced by NA.
...	Other parameters handed to the function lighten().

Details

The color adjustment can be calculated in three different color spaces.

1. If space = "HCL", the colors are transformed to HCL, (polarLUV), the luminance component L is adjusted, and then the colors are transformed back to a hexadecimal RGB string.
2. If space = "HLS", the colors are transformed to HLS, the lightness component L is adjusted, and then the color is transformed back to a hexadecimal RGB string.
3. If space = "combined", the colors are first adjusted in both the HCL and HLS spaces. Then, the adjusted HLS colors are converted into HCL, and then the chroma components of the adjusted HLS colors are copied to the adjusted HCL colors. Thus, in effect, the combined model adjusts luminance in HCL space but chroma in HLS space.

We have found that typically space = "HCL" performs best for lightening colors and space = "combined" performs best for darkening colors, and these are the default settings for lighten and darken, respectively.

Regardless of the chosen color space, the adjustment of the L component can occur by two methods, relative (the default) and absolute. Under the absolute method, the adjustment is L +/- 100 * amount when lightening/darkening colors. Under the relative method, the adjustment is 100 - (100 - L) * (1 - amount) when lightening colors and L * (1 - amount) when darkening colors.

Programmatically lightening and darkening colors can yield unexpected results (see examples). In HCL space, colors can become either too gray or overly colorful. By contrast, in HLS space it can happen that the overall amount of lightening or darkening appears to be non-uniform among a group of colors that are lightened or darkened jointly, and again, colors can become either too gray or overly colorful. We recommend to try different color spaces if the default space for the chosen function (lighten or darken) does not look right in a specific application.

Value

A character vector with (s)RGB codings of the colors in the palette.

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

polarLUV, hex, desaturate

Examples

```
# lighten dark colors, example 1
cl <- qualitative_hcl(5)
swatchplot(list(
  HCL = rbind("0%" = cl,
              "15%" = lighten(cl, 0.15),
              "30%" = lighten(cl, 0.3)),
  HLS = rbind("0%" = cl,
              "15%" = lighten(cl, 0.15, space = "HLS"),
              "30%" = lighten(cl, 0.3, space = "HLS")),
  combined = rbind("0%" = cl,
              "15%" = lighten(cl, 0.15, space = "combined"),
              "30%" = lighten(cl, 0.3, space = "combined"))),
  nrow = 4, line = 2.5
)

# lighten dark colors, example 2
cl <- c("#61A9D9", "#ADD668", "#E6D152", "#CE6BAF", "#797CBA")
swatchplot(list(
  HCL = rbind("0%" = cl,
              "15%" = lighten(cl, 0.15),
              "30%" = lighten(cl, 0.3)),
  HLS = rbind("0%" = cl,
              "15%" = lighten(cl, 0.15, space = "HLS"),
              "30%" = lighten(cl, 0.3, space = "HLS")),
  combined = rbind("0%" = cl,
              "15%" = lighten(cl, 0.15, space = "combined"),
              "30%" = lighten(cl, 0.3, space = "combined"))),
  nrow = 4, line = 2.5
)

# darken light colors, example 1
cl <- qualitative_hcl(5, "Pastel 1")
swatchplot(list(
  combined = rbind("0%" = cl,
              "15%" = darken(cl, 0.15),
              "30%" = darken(cl, 0.3)),
  HCL = rbind("0%" = cl,
              "15%" = darken(cl, 0.15, space = "HCL"),
              "30%" = darken(cl, 0.3, space = "HCL")),
  HLS = rbind("0%" = cl,
              "15%" = darken(cl, 0.15, space = "HLS"),
              "30%" = darken(cl, 0.3, space = "HLS"))),
  nrow = 4, line = 2.5
)

# darken light colors, example 2
cl <- c("#CDE4F3","#E7F3D3","#F7F0C7","#EFCFE5","#D0D1E7")
swatchplot(list(
  combined = rbind("0%" = cl,
              "15%" = darken(cl, 0.15),
              "30%" = darken(cl, 0.3)),
```

```
  HCL = rbind("0%" = cl,
              "15%" = darken(cl, 0.15, space = "HCL"),
              "30%" = darken(cl, 0.3, space = "HCL")),
  HLS = rbind("0%" = cl,
              "15%" = darken(cl, 0.15, space = "HLS"),
              "30%" = darken(cl, 0.3, space = "HLS"))),
  nrow = 4, line = 2.5
)
```

LUV *Create LUV Colors*

Description

This function creates colors of class "LUV"; a subclass of the virtual color-class class.

Usage

```
LUV(L, U, V, names)
```

Arguments

L, U, V	these arguments give the L, U and V coordinates of the colors. The values can be provided in separate L, U and V vectors or in a three-column matrix passed as L.
names	a vector of names for the colors (by default the row names of L are used).

Details

The L, U and V values give the coordinates of the colors in the CIE (1976) $L^*u^*v^*$ space. This is a transformation of the 1931 CIE XYZ space which attempts to produce perceptually based axes. Luminance takes values between 0 and 100, and the other coordinates typically take values between -100 and 100, although these values can also be exceeded by highly saturated colors. The u and v coordinates measure positions on green/red and blue/yellow axes.

Value

An object of class LUV which inherits from class color.

Author(s)

Ross Ihaka

See Also

RGB, HSV, XYZ, LAB, polarLAB, polarLUV.

Examples

```
## Show the LUV space
set.seed(1)
x <- sRGB(runif(1000), runif(1000), runif(1000))
y <- as(x, "LUV")
head(x)
head(y)
plot(y)
```

`max_chroma` *Compute Maximum Chroma for Given Hue and Luminance in HCL*

Description

Compute approximately the maximum chroma possible for a given hue and luminance combination in the HCL color space.

Usage

```
max_chroma(h, l, floor = FALSE)

max_chroma_table
```

Arguments

`h`	hue value in the HCL color description, has to be in [0, 360].
`l`	luminance value in the HCL color description, has to be in [0, 100].
`floor`	logical. Should the chroma value be rounded down to the next lower integer?

Details

As the possible combinations of chroma and luminance depend on hue, it is not obvious which maximum chroma can be used for a given combination of hue and luminance prior to calling `polarLUV`. To avoid having to `fixup` the color upon conversion to RGB `hex` codes, the `max_chroma` function computes (approximately) the maximum chroma possible. The computations are based on interpolations of pre-computed maxima in `max_chroma_table`, containing the maximum chroma for a given hue-luminance combination (both in integers). Hence, the result may sometimes still be very slightly larger than the actual maximum which can be avoided by taking the `floor` of the approximate value.

Value

A numeric vector with the maximum chroma coordinates.

See Also

`polarLUV`, `hex`

Examples

```
max_chroma(h = 0:36 * 10, l = 50)
max_chroma(h = 120, l = 0:10 * 10)
```

`mixcolor` *Compute the Convex Combination of Two Colors*

Description

This function can be used to compute the result of color mixing, assuming additive mixing (e.g., as appropriate for RGB or XYZ).

Usage

```
mixcolor(alpha, color1, color2, where = class(color1))
```

Arguments

`alpha`	The mixed color is obtained by combining an amount `1 - alpha` of `color1` with an amount `alpha` of `color2`.
`color1`	The first color.
`color2`	The second color.
`where`	The color space where the mixing is to take place.

Value

The mixed color. This is in the color space specified by `where`.

Author(s)

Ross Ihaka

See Also

`RGB`, `HSV`, `XYZ`, `LAB`, `polarLAB`, `LUV`, `polarLUV`.

Examples

```
mixcolor(0.5, sRGB(1, 0, 0), sRGB(0, 1, 0))
```

polarLAB *Create polarLAB Colors*

Description

This function creates colors of class "polarLAB"; a subclass of the virtual color-class class.

Usage

```
polarLAB(L, C, H, names)
```

Arguments

L, C, H	these arguments give the L, C and H coordinates of the colors. The values can be provided in separate L, C and H vectors or in a three-column matrix passed as L.
names	A vector of names for the colors (by default the row names of L are used).

Details

The polarLAB space is a transformation of the CIE $L^*a^*b^*$ space so that the a and b values are converted to polar coordinates. The radial component C measures chroma and the angular coordinate H is measures hue.

Value

An object of class polarLAB which inherits from class color.

Author(s)

Ross Ihaka

See Also

RGB, HSV, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
## Show the polarLAB space
set.seed(1)
x <- sRGB(runif(1000), runif(1000), runif(1000))
y <- as(x, "polarLAB")
head(x)
head(y)
plot(y)
```

polarLUV *Create polarLUV (HCL) Colors*

Description

This function creates colors of class "polarLUV"; a subclass of the virtual color-class class.

Usage

```
polarLUV(L, C, H, names)
```

Arguments

L, C, H	these arguments give the L, C and H coordinates of the colors. The values can be provided in separate L, C and H vectors or in a three-column matrix passed as L.
names	A vector of names for the colors (by default the row names of L are used).

Details

The polarLUV space is a transformation of the CIE $L^*u^*v^*$ space so that the u and v values are converted to polar coordinates. The radial component C measures chroma and the angular coordinate H is measures hue. It is also known as the HCL (hue-chroma-luminance) space.

Value

An object of class polarLUV which inherits from class color.

Author(s)

Ross Ihaka

See Also

RGB, HSV, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
## Show the polarLUV space
set.seed(1)
x <- sRGB(runif(1000), runif(1000), runif(1000))
y <- as(x, "polarLUV")
head(x)
head(y)
plot(y)
```

rainbow_hcl *HCL (and HSV) Color Palettes Corresponding to Base R Palettes*

Description

Color palettes based on the HCL (and HSV) color space to replace base R palettes.

Usage

```
rainbow_hcl(
  n,
  c = 50,
  l = 70,
  start = 0,
  end = 360 * (n - 1)/n,
  gamma = NULL,
  fixup = TRUE,
  alpha = 1,
  ...
)

heat_hcl(
  n,
  h = c(0, 90),
  c. = c(100, 30),
  l = c(50, 90),
  power = c(1/5, 1),
  gamma = NULL,
  fixup = TRUE,
  alpha = 1,
  ...
)

terrain_hcl(
  n,
  h = c(130, 0),
  c. = c(80, 0),
  l = c(60, 95),
  power = c(1/10, 1),
  gamma = NULL,
  fixup = TRUE,
  alpha = 1,
  ...
)

diverging_hsv(
  n,
```

```
  h = c(240, 0),
  s = 1,
  v = 1,
  power = 1,
  gamma = NULL,
  fixup = TRUE,
  alpha = 1,
  ...
)
```

Arguments

n	the number of colors (≥ 1) to be in the palette.
c, c.	chroma value in the HCL color description.
l	luminance value in the HCL color description.
start	the hue at which the rainbow begins.
end	the hue at which the rainbow ends.
gamma	Deprecated.
fixup	logical. Should the color be corrected to a valid RGB value before correction?
alpha	numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque).
...	Other arguments passed to hex.
h	hue value in the HCL or HSV color description, has to be in [0, 360] for HCL and in [0, 1] for HSV colors.
power	control parameter determining how chroma and luminance should be increased (1 = linear, 2 = quadratic, etc.).
s	saturation value in the HSV color description.
v	value value in the HSV color description.

Details

Based on the general qualitative, sequential, and diverging hcl_palettes within the colorspace package, convenience functions are provided as alternatives to standard base R palettes (which are highly saturated and too flashy).

rainbow_hcl computes a rainbow of colors via qualitative_hcl defined by different hues given a single value of each chroma and luminance. It corresponds to rainbow which computes a rainbow in HSV space.

heat_hcl is an implementation of heat.colors in HCL space based on a call to sequential_hcl. Similarly, terrain_hcl palette also calls sequential_hcl with different parameters, providing colors similar in spirit to terrain.colors in HCL space.

diverging_hsv (and equivalently its alias diverge_hsv) provides an HSV-based version of diverging_hcl. Its purpose is mainly didactic to show that HSV-based diverging palettes are less appealing, more difficult to read and more flashy than HCL-based diverging palettes. diverging_hsv is similar to cm.colors.

Value

A character vector with (s)RGB codings of the colors in the palette.

References

Zeileis A, Hornik K, Murrell P (2009). Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis*, **53**, 3259–3270. doi: 10.1016/j.csda.2008.11.033 Preprint available from https://www.zeileis.org/papers/Zeileis+Hornik+Murrell-2009.pdf.

Stauffer R, Mayr GJ, Dabernig M, Zeileis A (2015). Somewhere over the Rainbow: How to Make Effective Use of Colors in Meteorological Visualizations. *Bulletin of the American Meteorological Society*, **96**(2), 203–216. doi: 10.1175/BAMSD1300155.1

See Also

polarLUV, HSV, hex

Examples

```
## convenience demo function
wheel <- function(col, radius = 1, ...)
  pie(rep(1, length(col)), col = col, radius = radius, ...)

## compare base and colorspace palettes
## (in color and desaturated)
par(mar = rep(0, 4), mfrow = c(2, 2))
## rainbow color wheel
wheel(rainbow_hcl(12))
wheel(rainbow(12))
wheel(desaturate(rainbow_hcl(12)))
wheel(desaturate(rainbow(12)))

## diverging red-blue colors
swatchplot(
  diverging_hsv(7),
  desaturate(diverging_hsv(7)),
  diverging_hcl(7, c = 100, l = c(50, 90)),
  desaturate(diverging_hcl(7, c = 100, l = c(50, 90))),
  nrow = 2
)

## diverging cyan-magenta colors
swatchplot(
  cm.colors(7),
  desaturate(cm.colors(7)),
  diverging_hcl(7, "Cyan-Magenta"), ## or, similarly: Tropic
  desaturate(diverging_hcl(7, "Cyan-Magenta")),
  nrow = 2
)

## heat colors
```

```
swatchplot(
  heat.colors(12),
  desaturate(heat.colors(12)),
  heat_hcl(12),
  desaturate(heat_hcl(12)),
  nrow = 2
)

## terrain colors
swatchplot(
  terrain.colors(12),
  desaturate(terrain.colors(12)),
  terrain_hcl(12),
  desaturate(terrain_hcl(12)),
  nrow = 2
)
```

---

readhex *Read Hexadecimal Color Descriptions*

Description

This function reads a set of hexadecimal color descriptions from a file and creates a color object containing the corresponding colors.

Usage

```
readhex(file = "", class = "RGB")
```

Arguments

file	The file containing the color descriptions.
class	The kind of color object to be returned.

Details

The file is assumed to contain hexadecimal color descriptions of the form #RRGGBB.

Value

An color object of the specified class containing the color descriptions.

Author(s)

Ross Ihaka

See Also

writehex, readRGB, hex2RGB, RGB, HSV, XYZ, LAB, polarLAB, LUV, polarLUV,

Examples

```
## Not run:
rgb <- readhex("pastel.txt")
hsv <- readhex("pastel.txt", "HSV")

## End(Not run)
```

---

readRGB *Read RGB Color Descriptions*

Description

This function reads a set of RGB color descriptions (of the form written by `gcolorsel`) from a file and creates a color object containing the corresponding colors.

Usage

```
readRGB(file, class = "RGB")
```

Arguments

`file`	The file containing the color descriptions.
`class`	The kind of color object to be returned.

Details

The file is assumed to contain RGB color descriptions consisting of three integer values in the range from 0 to 255 followed by a color name.

Value

An color object of the specified class containing the color descriptions.

Author(s)

Ross Ihaka

See Also

`writehex`, `readhex`, `hex2RGB`, `RGB`, `HSV`, `XYZ`, `LAB`, `polarLAB`, `LUV`, `polarLUV`.

Examples

```
## Not run:
rgb <- readRGB("pastel.rgb")
hsv <- readRGB("pastel.rgb", "HSV")

## End(Not run)
```

RGB *Create RGB Colors*

Description

This function creates colors of class RGB; a subclass of the virtual color-class class.

Usage

```
RGB(R, G, B, names)
```

Arguments

R, G, B	these arguments give the red, green and blue intensities of the colors (the values should lie between 0 and 1). The values can be provided in separate R, G and B vectors or in a three-column matrix passed as R.
names	A vector of names for the colors (by default the row names of R are used).

Details

This function creates colors in the linearized sRGB color space (IEC standard 61966).

Value

An object of class RGB which inherits from class color.

Author(s)

Ross Ihaka

See Also

sRGB, HSV, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
# Create a random set of colors
set.seed(1)
RGB(R = runif(20), G = runif(20), B = runif(20))
```

scale_colour_binned_diverging *HCL-Based Binned Diverging Color Scales for ggplot2*

Description

Binned ggplot2 color scales using the color palettes generated by diverging_hcl.

Usage

```
scale_colour_binned_diverging(
  palette = NULL,
  c1 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = FALSE,
  mid = 0,
  na.value = "grey50",
  guide = "coloursteps",
  n_interp = 11,
  aesthetics = "colour",
  ...
)

scale_color_binned_diverging(
  palette = NULL,
  c1 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = FALSE,
  mid = 0,
  na.value = "grey50",
  guide = "coloursteps",
  n_interp = 11,
  aesthetics = "colour",
```

```
  ...
)

scale_fill_binned_diverging(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run `hcl_palettes(type = "diverging")` for available options.
c1	Chroma value at the scale endpoints.
cmax	Maximum chroma value.
l1	Luminance value at the scale endpoints.
l2	Luminance value at the scale midpoint.
h1	Hue value at the first endpoint.
h2	Hue value at the second endpoint.
p1	Control parameter determining how chroma should vary (1 = linear, 2 = quadratic, etc.).
p2	Control parameter determining how luminance should vary (1 = linear, 2 = quadratic, etc.).
alpha	Numeric vector of values in the range `[0, 1]` for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If `TRUE`, reverses the order of the colors in the color scale.
mid	Data value that should be mapped to the mid-point of the diverging color scale.
na.value	Color to be used for missing data points.
guide	Type of legend. Use `"coloursteps"` for color bar with discrete steps.
n_interp	Number of discrete colors that should be used to interpolate the binned color scale. It is important to use an odd number to capture the color at the midpoint.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
...	common continuous scale parameters: 'name', 'breaks', 'labels', and 'limits'. See `binned_scale` for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
# adapted from stackoverflow: https://stackoverflow.com/a/20127706/4975218

library("ggplot2")

# generate dataset and base plot
set.seed(100)
df <- data.frame(country = LETTERS, V = runif(26, -40, 40))
```

```
df$country = factor(LETTERS, LETTERS[order(df$V)]) # reorder factors
gg <- ggplot(df, aes(x = country, y = V, fill = V)) +
  geom_bar(stat = "identity") +
  labs(y = "Under/over valuation in %", x = "Country") +
  coord_flip() + theme_minimal()

# plot with default diverging scale
gg + scale_fill_binned_diverging(n.breaks = 6)

# plot with alternative scale
gg + scale_fill_binned_diverging(palette = "Purple-Green", n.breaks = 6)
```

`scale_colour_binned_divergingx` *HCL-Based Binned Flexible Diverging Scales for ggplot2*

Description

Binned ggplot2 color scales using the color palettes generated by `divergingx_hcl`.

Usage

```
scale_colour_binned_divergingx(
  palette = "Geyser",
  c1 = NULL,
  c2 = NULL,
  c3 = NULL,
  l1 = NULL,
  l2 = NULL,
  l3 = NULL,
  h1 = NULL,
  h2 = NULL,
  h3 = NULL,
  p1 = NULL,
  p2 = NULL,
  p3 = NULL,
  p4 = NULL,
  cmax1 = NULL,
  cmax2 = NULL,
  alpha = 1,
  rev = FALSE,
  mid = 0,
  na.value = "grey50",
  guide = "coloursteps",
  n_interp = 11,
  aesthetics = "colour",
  ...
)
```

```
scale_color_binned_divergingx(
  palette = "Geyser",
  c1 = NULL,
  c2 = NULL,
  c3 = NULL,
  l1 = NULL,
  l2 = NULL,
  l3 = NULL,
  h1 = NULL,
  h2 = NULL,
  h3 = NULL,
  p1 = NULL,
  p2 = NULL,
  p3 = NULL,
  p4 = NULL,
  cmax1 = NULL,
  cmax2 = NULL,
  alpha = 1,
  rev = FALSE,
  mid = 0,
  na.value = "grey50",
  guide = "coloursteps",
  n_interp = 11,
  aesthetics = "colour",
  ...
)

scale_fill_binned_divergingx(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used.
h1, h2, h3, c1, c2, c3, l1, l2, l3, p1, p2, p3, p4, cmax1, cmax2	Parameters to customize the scale. See divergingx_hcl for details.
alpha	Numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If TRUE, reverses the order of the colors in the color scale.
mid	Data value that should be mapped to the mid-point of the diverging color scale.
na.value	Color to be used for missing data points.
guide	Type of legend. Use "coloursteps" for color bar with discrete steps.
n_interp	Number of discrete colors that should be used to interpolate the binned color scale. For diverging scales, it is important to use an odd number to capture the color at the midpoint.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
...	common binned scale parameters: 'name', 'breaks', 'labels', and 'limits'. See binned_scale for more details.

Details

Available CARTO palettes: ArmyRose, Earth, Fall, Geyser, TealRose, Temps, Tropic.

Available ColorBrewer.org palettes: Spectral, PuOr, RdYlGn, RdYlBu, RdGy, BrBG, PiYG, PRGn, RdBu.

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
library("ggplot2")

# volcano plot (difference from mean height)
nx = 87
ny = 61
df <- data.frame(diff = c(volcano) - mean(volcano), x = rep(1:nx, ny), y = rep(1:ny, each = nx))
ggplot(df, aes(x, y, fill=diff)) +
  geom_raster() + scale_fill_binned_divergingx(palette = "Fall", rev = TRUE) +
  coord_fixed(expand = FALSE)


# adapted from stackoverflow: https://stackoverflow.com/a/20127706/4975218

# generate dataset and base plot
set.seed(100)
df <- data.frame(country = LETTERS, V = runif(26, -40, 40))
df$country = factor(LETTERS, LETTERS[order(df$V)]) # reorder factors
gg <- ggplot(df, aes(x = country, y = V, fill = V)) +
  geom_bar(stat = "identity") +
  labs(y = "Under/over valuation in %", x = "Country") +
  coord_flip() + theme_minimal()

# plot with diverging scale "Geyser"
gg + scale_fill_binned_divergingx(palette = "Geyser",  n.breaks = 6)
```

---

scale_colour_binned_qualitative

HCL-Based Binned Qualitative Color Scales for ggplot2

Description

Binned ggplot2 color scales using the color palettes generated by `qualitative_hcl`. These scales are provided for completeness. It is not normally a good idea to color a continuous, binned variable using a qualitative scale.

Usage

```
scale_colour_binned_qualitative(
  palette = NULL,
  c1 = NULL,
  l1 = NULL,
  h1 = NULL,
  h2 = NULL,
  alpha = 1,
  rev = FALSE,
  begin = 0,
  end = 1,
  na.value = "grey50",
  guide = "coloursteps",
  aesthetics = "colour",
  n_interp = 11,
  ...
)

scale_color_binned_qualitative(
  palette = NULL,
  c1 = NULL,
  l1 = NULL,
  h1 = NULL,
  h2 = NULL,
  alpha = 1,
  rev = FALSE,
  begin = 0,
  end = 1,
  na.value = "grey50",
  guide = "coloursteps",
  aesthetics = "colour",
  n_interp = 11,
  ...
)

scale_fill_binned_qualitative(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run `hcl_palettes(type = "qualitative")` for available options.
c1	Chroma value, used for all colors in the scale.
l1	Luminance value, used for all colors in the scale.
h1	Beginning hue value.
h2	Ending hue value.
alpha	Numeric vector of values in the range `[0, 1]` for alpha transparency channel (0 means transparent and 1 means opaque).

rev	If TRUE, reverses the order of the colors in the color scale.
begin	Number in the range of [0, 1] indicating to which point in the color scale the smallest data value should be mapped.
end	Number in the range of [0, 1] indicating to which point in the color scale the largest data value should be mapped.
na.value	Color to be used for missing data points.
guide	Type of legend. Use "coloursteps" for color bar with discrete steps.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
n_interp	Number of discrete colors that should be used to interpolate the binned color scale. 11 will work fine in most cases.
...	common binned scale parameters: 'name', 'breaks', 'labels', and 'limits'. See binned_scale for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
library("ggplot2")

# none of these examples are necessarily good ideas
gg <- ggplot(iris, aes(x = Species, y = Sepal.Width, color = Sepal.Length)) +
  geom_jitter(width = 0.3) + theme_minimal()

gg + scale_color_binned_qualitative(palette = "Dynamic")
gg + scale_color_binned_qualitative(palette = "Dark3", l1 = 70)

nx = 87
ny = 61
df <- data.frame(height = c(volcano), x = rep(1:nx, ny), y = rep(1:ny, each = nx))
ggplot(df, aes(x, y, fill=height)) +
  geom_raster() + scale_fill_binned_qualitative(palette = "Dark 3") +
  coord_fixed(expand = FALSE)
```

---

scale_colour_binned_sequential

HCL-Based Binned Sequential Color Scales for ggplot2

Description

Binned ggplot2 color scales using the color palettes generated by sequential_hcl.

Usage

```
scale_colour_binned_sequential(
  palette = NULL,
  c1 = NULL,
  c2 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = TRUE,
  begin = 0,
  end = 1,
  na.value = "grey50",
  guide = "coloursteps",
  aesthetics = "colour",
  n_interp = 11,
  ...
)

scale_color_binned_sequential(
  palette = NULL,
  c1 = NULL,
  c2 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = TRUE,
  begin = 0,
  end = 1,
  na.value = "grey50",
  guide = "coloursteps",
  aesthetics = "colour",
  n_interp = 11,
  ...
)

scale_fill_binned_sequential(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run `hcl_palettes(type = "sequential")` for available options.
c1	Beginning chroma value.
c2	Ending chroma value.
cmax	Maximum chroma value.
l1	Beginning luminance value.
l2	Ending luminance value.
h1	Beginning hue value.
h2	Ending hue value. If set to `NA`, generates a single-hue scale.
p1	Control parameter determining how chroma should vary (1 = linear, 2 = quadratic, etc.).
p2	Control parameter determining how luminance should vary (1 = linear, 2 = quadratic, etc.).
alpha	Numeric vector of values in the range `[0, 1]` for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If `TRUE` (default), reverses the order of the colors in the color scale (compared to `sequential_hcl`).
begin	Number in the range of `[0, 1]` indicating to which point in the color scale the smallest data value should be mapped.
end	Number in the range of `[0, 1]` indicating to which point in the color scale the largest data value should be mapped.
na.value	Color to be used for missing data points.
guide	Type of legend. Use `"coloursteps"` for color bar with discrete steps.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
n_interp	Number of discrete colors that should be used to interpolate the binned color scale. 11 will work fine in most cases.
...	common binned scale parameters: ‘name‘, ‘breaks‘, ‘labels‘, and ‘limits‘. See `binned_scale` for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Compared to `sequential_hcl` the ordering of the colors in the sequential ggplot2 scale are reversed by default (i.e., `rev = TRUE`) to be more consistent with ggplot2's own scales such as `scale_color_fermenter`. For most named palettes this leads to darker and more colorful colors for larger values on the scale. This is typically the better default on light/white backgrounds.

Examples

```
library("ggplot2")

# volcano plot
df <- data.frame(height = c(volcano), x = c(row(volcano)), y = c(col(volcano)))
ggplot(df, aes(x, y, fill = height)) +
  geom_raster() + scale_fill_binned_sequential(palette = "Terrain", rev = FALSE) +
  coord_fixed(expand = FALSE)
```

---

scale_colour_continuous_diverging

HCL-Based Continuous Diverging Color Scales for ggplot2

Description

Continuous ggplot2 color scales using the color palettes generated by diverging_hcl.

Usage

```
scale_colour_continuous_diverging(
  palette = NULL,
  c1 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = FALSE,
  mid = 0,
  na.value = "grey50",
  guide = "colourbar",
  n_interp = 11,
  aesthetics = "colour",
  ...
)

scale_color_continuous_diverging(
  palette = NULL,
  c1 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
```

```
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = FALSE,
  mid = 0,
  na.value = "grey50",
  guide = "colourbar",
  n_interp = 11,
  aesthetics = "colour",
  ...
)

scale_fill_continuous_diverging(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run `hcl_palettes(type = "diverging")` for available options.
c1	Chroma value at the scale endpoints.
cmax	Maximum chroma value.
l1	Luminance value at the scale endpoints.
l2	Luminance value at the scale midpoint.
h1	Hue value at the first endpoint.
h2	Hue value at the second endpoint.
p1	Control parameter determining how chroma should vary (1 = linear, 2 = quadratic, etc.).
p2	Control parameter determining how luminance should vary (1 = linear, 2 = quadratic, etc.).
alpha	Numeric vector of values in the range `[0, 1]` for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If `TRUE`, reverses the order of the colors in the color scale.
mid	Data value that should be mapped to the mid-point of the diverging color scale.
na.value	Color to be used for missing data points.
guide	Type of legend. Use `"colourbar"` for continuous color bar.
n_interp	Number of discrete colors that should be used to interpolate the continuous color scale. It is important to use an odd number to capture the color at the midpoint.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
...	common continuous scale parameters: 'name', 'breaks', 'labels', and 'limits'. See `continuous_scale` for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
# adapted from stackoverflow: https://stackoverflow.com/a/20127706/4975218

library("ggplot2")

# generate dataset and base plot
set.seed(100)
df <- data.frame(country = LETTERS, V = runif(26, -40, 40))
df$country = factor(LETTERS, LETTERS[order(df$V)]) # reorder factors
gg <- ggplot(df, aes(x = country, y = V, fill = V)) +
  geom_bar(stat = "identity") +
  labs(y = "Under/over valuation in %", x = "Country") +
  coord_flip() + theme_minimal()

# plot with default diverging scale
gg + scale_fill_continuous_diverging()

# plot with alternative scale
gg + scale_fill_continuous_diverging(palette = "Purple-Green")

# plot with modified alternative scale
gg + scale_fill_continuous_diverging(palette = "Blue-Red 3", l1 = 30, l2 = 100, p1 = .9, p2 = 1.2)
```

---

scale_colour_continuous_divergingx

HCL-Based Continuous Flexible Diverging Scales for ggplot2

Description

Continuous ggplot2 color scales using the color palettes generated by divergingx_hcl.

Usage

```
scale_colour_continuous_divergingx(
  palette = "Geyser",
  c1 = NULL,
  c2 = NULL,
  c3 = NULL,
  l1 = NULL,
  l2 = NULL,
  l3 = NULL,
  h1 = NULL,
  h2 = NULL,
  h3 = NULL,
  p1 = NULL,
  p2 = NULL,
  p3 = NULL,
  p4 = NULL,
```

```
  cmax1 = NULL,
  cmax2 = NULL,
  alpha = 1,
  rev = FALSE,
  mid = 0,
  na.value = "grey50",
  guide = "colourbar",
  n_interp = 11,
  aesthetics = "colour",
  ...
)

scale_color_continuous_divergingx(
  palette = "Geyser",
  c1 = NULL,
  c2 = NULL,
  c3 = NULL,
  l1 = NULL,
  l2 = NULL,
  l3 = NULL,
  h1 = NULL,
  h2 = NULL,
  h3 = NULL,
  p1 = NULL,
  p2 = NULL,
  p3 = NULL,
  p4 = NULL,
  cmax1 = NULL,
  cmax2 = NULL,
  alpha = 1,
  rev = FALSE,
  mid = 0,
  na.value = "grey50",
  guide = "colourbar",
  n_interp = 11,
  aesthetics = "colour",
  ...
)

scale_fill_continuous_divergingx(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used.
h1, h2, h3, c1, c2, c3, l1, l2, l3, p1, p2, p3, p4, cmax1, cmax2	Parameters to customize the scale. See divergingx_hcl for details.
alpha	Numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque).

rev	If TRUE, reverses the order of the colors in the color scale.
mid	Data value that should be mapped to the mid-point of the diverging color scale.
na.value	Color to be used for missing data points.
guide	Type of legend. Use "colourbar" for continuous color bar.
n_interp	Number of discrete colors that should be used to interpolate the continuous color scale. For diverging scales, it is important to use an odd number to capture the color at the midpoint.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
...	common continuous scale parameters: 'name', 'breaks', 'labels', and 'limits'. See continuous_scale for more details.

Details

Available CARTO palettes: ArmyRose, Earth, Fall, Geyser, TealRose, Temps, Tropic.

Available ColorBrewer.org palettes: Spectral, PuOr, RdYlGn, RdYlBu, RdGy, BrBG, PiYG, PRGn, RdBu.

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
library("ggplot2")

# volcano plot (difference from mean height)
nx = 87
ny = 61
df <- data.frame(diff = c(volcano) - mean(volcano), x = rep(1:nx, ny), y = rep(1:ny, each = nx))
ggplot(df, aes(x, y, fill=diff)) +
  geom_raster() + scale_fill_continuous_divergingx(palette = "Fall", rev = TRUE) +
  coord_fixed(expand = FALSE)


# adapted from stackoverflow: https://stackoverflow.com/a/20127706/4975218

# generate dataset and base plot
set.seed(100)
df <- data.frame(country = LETTERS, V = runif(26, -40, 40))
df$country = factor(LETTERS, LETTERS[order(df$V)]) # reorder factors
gg <- ggplot(df, aes(x = country, y = V, fill = V)) +
  geom_bar(stat = "identity") +
  labs(y = "Under/over valuation in %", x = "Country") +
  coord_flip() + theme_minimal()

# plot with diverging scale "Geyser"
gg + scale_fill_continuous_divergingx(palette = "Geyser")

# plot with diverging scale "ArmyRose"
gg + scale_fill_continuous_divergingx(palette = "ArmyRose")
```

scale_colour_continuous_qualitative *HCL-Based Continuous Qualitative Color Scales for ggplot2*

Description

Continuous ggplot2 color scales using the color palettes generated by qualitative_hcl. These scales are provided for completeness. It is not normally a good idea to color a continuous variable using a qualitative scale.

Usage

```
scale_colour_continuous_qualitative(
  palette = NULL,
  c1 = NULL,
  l1 = NULL,
  h1 = NULL,
  h2 = NULL,
  alpha = 1,
  rev = FALSE,
  begin = 0,
  end = 1,
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour",
  n_interp = 11,
  ...
)

scale_color_continuous_qualitative(
  palette = NULL,
  c1 = NULL,
  l1 = NULL,
  h1 = NULL,
  h2 = NULL,
  alpha = 1,
  rev = FALSE,
  begin = 0,
  end = 1,
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour",
  n_interp = 11,
  ...
)

scale_fill_continuous_qualitative(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run `hcl_palettes(type = "qualitative")` for available options.
c1	Chroma value, used for all colors in the scale.
l1	Luminance value, used for all colors in the scale.
h1	Beginning hue value.
h2	Ending hue value.
alpha	Numeric vector of values in the range `[0, 1]` for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If `TRUE`, reverses the order of the colors in the color scale.
begin	Number in the range of `[0, 1]` indicating to which point in the color scale the smallest data value should be mapped.
end	Number in the range of `[0, 1]` indicating to which point in the color scale the largest data value should be mapped.
na.value	Color to be used for missing data points.
guide	Type of legend. Use `"colourbar"` for continuous color bar.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
n_interp	Number of discrete colors that should be used to interpolate the continuous color scale. 11 will work fine in most cases.
...	common continuous scale parameters: 'name', 'breaks', 'labels', and 'limits'. See `continuous_scale` for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
library("ggplot2")

# none of these examples are necessarily good ideas
gg <- ggplot(iris, aes(x = Species, y = Sepal.Width, color = Sepal.Length)) +
  geom_jitter(width = 0.3) + theme_minimal()

gg + scale_color_continuous_qualitative(palette = "Warm")
gg + scale_color_continuous_qualitative(palette = "Cold", l1 = 60)
gg + scale_color_continuous_qualitative(palette = "Harmonic", rev = TRUE)

nx = 87
ny = 61
df <- data.frame(height = c(volcano), x = rep(1:nx, ny), y = rep(1:ny, each = nx))
ggplot(df, aes(x, y, fill=height)) +
  geom_raster() + scale_fill_continuous_qualitative(palette = "Dark 3") +
  coord_fixed(expand = FALSE)
```

scale_colour_continuous_sequential *HCL-Based Continuous Sequential Color Scales for ggplot2*

Description

Continuous ggplot2 color scales using the color palettes generated by sequential_hcl.

Usage

```
scale_colour_continuous_sequential(
  palette = NULL,
  c1 = NULL,
  c2 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = TRUE,
  begin = 0,
  end = 1,
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour",
  n_interp = 11,
  ...
)

scale_color_continuous_sequential(
  palette = NULL,
  c1 = NULL,
  c2 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = TRUE,
  begin = 0,
  end = 1,
```

```
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour",
  n_interp = 11,
  ...
)

scale_fill_continuous_sequential(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run `hcl_palettes(type = "sequential")` for available options.
c1	Beginning chroma value.
c2	Ending chroma value.
cmax	Maximum chroma value.
l1	Beginning luminance value.
l2	Ending luminance value.
h1	Beginning hue value.
h2	Ending hue value. If set to `NA`, generates a single-hue scale.
p1	Control parameter determining how chroma should vary (1 = linear, 2 = quadratic, etc.).
p2	Control parameter determining how luminance should vary (1 = linear, 2 = quadratic, etc.).
alpha	Numeric vector of values in the range `[0, 1]` for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If `TRUE` (default), reverses the order of the colors in the color scale (compared to `sequential_hcl`).
begin	Number in the range of `[0, 1]` indicating to which point in the color scale the smallest data value should be mapped.
end	Number in the range of `[0, 1]` indicating to which point in the color scale the largest data value should be mapped.
na.value	Color to be used for missing data points.
guide	Type of legend. Use `"colourbar"` for continuous color bar.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
n_interp	Number of discrete colors that should be used to interpolate the continuous color scale. 11 will work fine in most cases.
...	common continuous scale parameters: ‘name‘, ‘breaks‘, ‘labels‘, and ‘limits‘. See `continuous_scale` for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Compared to `sequential_hcl` the ordering of the colors in the sequential ggplot2 scale are reversed by default (i.e., `rev = TRUE`) to be more consistent with ggplot2's own scales such as `scale_color_brewer`. For most named palettes this leads to darker and more colorful colors for larger values on the scale. This is typically the better default on light/white backgrounds.

Examples

```
library("ggplot2")

# base plot
gg <- ggplot(iris, aes(x = Species, y = Sepal.Width, color = Sepal.Length)) +
  geom_jitter(width = 0.3) + theme_minimal()

# default settings
gg + scale_color_continuous_sequential()

# switch palette and overwrite some default values
gg + scale_color_continuous_sequential(palette = "Reds", l1 = 20, c2 = 70, p1 = 1)

# select a range out of the entire palette
gg + scale_color_continuous_sequential(palette = "Heat", begin = 0.2, end = 0.8)

# volcano plot
df <- data.frame(height = c(volcano), x = c(row(volcano)), y = c(col(volcano)))
ggplot(df, aes(x, y, fill = height)) +
  geom_raster() + scale_fill_continuous_sequential(palette = "Terrain", rev = FALSE) +
  coord_fixed(expand = FALSE)
```

---

scale_colour_discrete_diverging

HCL-Based Discrete Diverging Color Scales for ggplot2

Description

Discrete ggplot2 color scales using the color palettes generated by `diverging_hcl`.

Usage

```
scale_colour_discrete_diverging(
  palette = NULL,
  c1 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
```

```
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = FALSE,
  nmax = NULL,
  order = NULL,
  aesthetics = "colour",
  ...
)

scale_color_discrete_diverging(
  palette = NULL,
  c1 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = FALSE,
  nmax = NULL,
  order = NULL,
  aesthetics = "colour",
  ...
)

scale_fill_discrete_diverging(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run `hcl_palettes(type = "diverging")` for available options.
c1	Chroma value at the scale endpoints.
cmax	Maximum chroma value.
l1	Luminance value at the scale endpoints.
l2	Luminance value at the scale midpoint.
h1	Hue value at the first endpoint.
h2	Hue value at the second endpoint.
p1	Control parameter determining how chroma should vary (1 = linear, 2 = quadratic, etc.).
p2	Control parameter determining how luminance should vary (1 = linear, 2 = quadratic, etc.).
alpha	Numeric vector of values in the range `[0, 1]` for alpha transparency channel (0 means transparent and 1 means opaque).

rev	If TRUE, reverses the order of the colors in the color scale.
nmax	Maximum number of different colors the palette should contain. If not provided, is calculated automatically from the data.
order	Numeric vector listing the order in which the colors should be used. Default is 1:nmax.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
...	common discrete scale parameters: name, breaks, labels, na.value, limits and guide. See discrete_scale for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
library("ggplot2")

# default colors with slightly darkened midpoint
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() + theme_minimal() +
  scale_color_discrete_diverging(l2=75)

# color scale "Green-Orange"
ggplot(iris, aes(Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.7) + theme_classic() +
    scale_fill_discrete_diverging(palette = "Green-Orange", rev = TRUE)

# use `nmax` and `order` to skip some colors
ggplot(iris, aes(Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.7) + theme_classic() +
    scale_fill_discrete_diverging(palette = "Green-Orange", nmax = 5, order = c(1, 4, 5))
```

---

scale_colour_discrete_divergingx

HCL-Based Discrete Flexible Diverging Scales for ggplot2

Description

Discrete ggplot2 color scales using the color palettes generated by divergingx_hcl.

Usage

```
scale_colour_discrete_divergingx(
  palette = "Geyser",
  c1 = NULL,
  c2 = NULL,
```

```
  c3 = NULL,
  l1 = NULL,
  l2 = NULL,
  l3 = NULL,
  h1 = NULL,
  h2 = NULL,
  h3 = NULL,
  p1 = NULL,
  p2 = NULL,
  p3 = NULL,
  p4 = NULL,
  cmax1 = NULL,
  cmax2 = NULL,
  alpha = 1,
  rev = FALSE,
  nmax = NULL,
  order = NULL,
  aesthetics = "colour",
  ...
)

scale_color_discrete_divergingx(
  palette = "Geyser",
  c1 = NULL,
  c2 = NULL,
  c3 = NULL,
  l1 = NULL,
  l2 = NULL,
  l3 = NULL,
  h1 = NULL,
  h2 = NULL,
  h3 = NULL,
  p1 = NULL,
  p2 = NULL,
  p3 = NULL,
  p4 = NULL,
  cmax1 = NULL,
  cmax2 = NULL,
  alpha = 1,
  rev = FALSE,
  nmax = NULL,
  order = NULL,
  aesthetics = "colour",
  ...
)

scale_fill_discrete_divergingx(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used.
h1, h2, h3, c1, c2, c3, l1, l2, l3, p1, p2, p3, p4, cmax1, cmax2	Parameters to customize the scale. See divergingx_hcl for details.
alpha	Numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If TRUE, reverses the order of the colors in the color scale.
nmax	Maximum number of different colors the palette should contain. If not provided, is calculated automatically from the data.
order	Numeric vector listing the order in which the colors should be used. Default is 1:nmax.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
...	common discrete scale parameters: name, breaks, labels, na.value, limits and guide. See discrete_scale for more details.

Details

Available CARTO palettes: ArmyRose, Earth, Fall, Geyser, TealRose, Temps, Tropic.

Available ColorBrewer.org palettes: Spectral, PuOr, RdYlGn, RdYlBu, RdGy, BrBG, PiYG, PRGn, RdBu.

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
library("ggplot2")

# default color scale
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() + theme_minimal() +
  scale_color_discrete_divergingx()

# color scale "Tropic"
ggplot(iris, aes(Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.7) + theme_classic() +
    scale_fill_discrete_divergingx(palette = "Tropic", rev = TRUE)

# use `nmax` and `order` to skip some colors
ggplot(iris, aes(Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.7) + theme_classic() +
    scale_fill_discrete_divergingx(palette = "Tropic", nmax = 5, order = c(1, 4, 5))
```

scale_colour_discrete_qualitative
HCL-Based Discrete Qualitative Color Scales for ggplot2

Description

Discrete ggplot2 color scales using the color palettes generated by qualitative_hcl.

Usage

```
scale_colour_discrete_qualitative(
  palette = NULL,
  c1 = NULL,
  l1 = NULL,
  h1 = NULL,
  h2 = NULL,
  alpha = 1,
  rev = FALSE,
  nmax = NULL,
  order = NULL,
  aesthetics = "colour",
  ...
)

scale_color_discrete_qualitative(
  palette = NULL,
  c1 = NULL,
  l1 = NULL,
  h1 = NULL,
  h2 = NULL,
  alpha = 1,
  rev = FALSE,
  nmax = NULL,
  order = NULL,
  aesthetics = "colour",
  ...
)

scale_fill_discrete_qualitative(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run hcl_palettes(type = "qualitative") for available options.
c1	Chroma value, used for all colors in the scale.
l1	Luminance value, used for all colors in the scale.

h1	Beginning hue value.
h2	Ending hue value.
alpha	Numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If TRUE, reverses the order of the colors in the color scale.
nmax	Maximum number of different colors the palette should contain. If not provided, is calculated automatically from the data.
order	Numeric vector listing the order in which the colors should be used. Default is 1:nmax.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
...	common discrete scale parameters: name, breaks, labels, na.value, limits and guide. See discrete_scale for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Examples

```
library("ggplot2")

# default colors
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() + theme_minimal() +
  scale_color_discrete_qualitative()

# color scale "Harmonic"
ggplot(iris, aes(Sepal.Length, fill = Species)) +
  geom_density(alpha = 0.7) + scale_fill_discrete_qualitative(palette = "Harmonic")
```

---

scale_colour_discrete_sequential

HCL-Based Discrete Sequential Color Scales for ggplot2

Description

Discrete ggplot2 color scales using the color palettes generated by sequential_hcl.

Usage

```
scale_colour_discrete_sequential(
  palette = NULL,
  c1 = NULL,
  c2 = NULL,
  cmax = NULL,
```

```
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = TRUE,
  nmax = NULL,
  order = NULL,
  aesthetics = "colour",
  ...
)

scale_color_discrete_sequential(
  palette = NULL,
  c1 = NULL,
  c2 = NULL,
  cmax = NULL,
  l1 = NULL,
  l2 = NULL,
  h1 = NULL,
  h2 = NULL,
  p1 = NULL,
  p2 = NULL,
  alpha = 1,
  rev = TRUE,
  nmax = NULL,
  order = NULL,
  aesthetics = "colour",
  ...
)

scale_fill_discrete_sequential(..., aesthetics = "fill")
```

Arguments

palette	The name of the palette to be used. Run `hcl_palettes(type = "sequential")` for available options.
c1	Beginning chroma value.
c2	Ending chroma value.
cmax	Maximum chroma value.
l1	Beginning luminance value.
l2	Ending luminance value.
h1	Beginning hue value.
h2	Ending hue value. If set to `NA`, generates a single-hue scale.

p1	Control parameter determining how chroma should vary (1 = linear, 2 = quadratic, etc.).
p2	Control parameter determining how luminance should vary (1 = linear, 2 = quadratic, etc.).
alpha	Numeric vector of values in the range [0, 1] for alpha transparency channel (0 means transparent and 1 means opaque).
rev	If TRUE (default), reverses the order of the colors in the color scale (compared to sequential_hcl).
nmax	Maximum number of different colors the palette should contain. If not provided, is calculated automatically from the data.
order	Numeric vector listing the order in which the colors should be used. Default is 1:nmax.
aesthetics	The ggplot2 aesthetics to which this scale should be applied.
...	common discrete scale parameters: name, breaks, labels, na.value, limits and guide. See discrete_scale for more details.

Details

If both a valid palette name and palette parameters are provided then the provided palette parameters overwrite the parameters in the named palette. This enables easy customization of named palettes.

Compared to sequential_hcl the ordering of the colors in the sequential ggplot2 scale are reversed by default (i.e., rev = TRUE) to be more consistent with ggplot2's own scales such as scale_color_brewer. For most named palettes this leads to darker and more colorful colors for larger values on the scale. This is typically the better default on light/white backgrounds.

Examples

```
library("ggplot2")

# default colors
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() + scale_color_discrete_sequential() + theme_classic()

# customization of named palette
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() + scale_colour_discrete_sequential(palette = "Reds", nmax = 4, p2 = 1.5) +
  theme_classic()

# color scale "Terrain"
ggplot(iris, aes(Sepal.Length, fill = Species)) +
 geom_density(alpha = 0.7) + scale_fill_discrete_sequential(palette = "Terrain") + theme_minimal()
```

`simulate_cvd` *Simulate Color Vision Deficiency*

Description

Transformation of R colors by simulating color vision deficiencies, based on a CVD transform matrix.

Usage

```
simulate_cvd(col, cvd_transform)

deutan(col, severity = 1)

protan(col, severity = 1)

tritan(col, severity = 1)

interpolate_cvd_transform(cvd, severity = 1)
```

Arguments

`col`	character. A color or vector of colors, e.g., `"#FFA801"` or `"blue"`. Input `col` can also be a matrix with three rows containing R/G/B (0-255) values, see details.
`cvd_transform`	numeric 3x3 matrix, specifying the color vision deficiency transform matrix.
`severity`	numeric. Severity of the color vision defect, a number between 0 and 1.
`cvd`	list of cvd transformation matrices. See `cvd` for available options.

Details

Using the physiologically-based model for simulating color vision deficiency (CVD) of Machado et al. (2009), different kinds of limitations can be emulated: deuteranope (green cone cells defective), protanope (red cone cells defective), and tritanope (blue cone cells defective). The workhorse function to do so is `simulate_cvd` which can take any vector of valid R colors and transform them according to a certain CVD transformation matrix (see `cvd`) and transformation equation.

The functions `deutan`, `protan`, and `tritan` are the high-level functions for simulating the corresponding kind of colorblindness with a given severity. Internally, they all call `simulate_cvd` along with a (possibly interpolated) version of the matrices from `cvd`. Matrix interpolation can be carried out with the function `interpolate_cvd_transform` (see Examples).

If input `col` is a matrix with three rows named `R`, `G`, and `B` (top down) they are interpreted as Red-Green-Blue values within the range `[0-255]`. Instead of an (s)RGB color vector a matrix of the same size as the input `col` with the corresponding simulated Red-Green-Blue values will be returned. This can be handy to avoid too many conversions.

References

Machado GM, Oliveira MM, Fernandes LAF (2009). A Physiologically-Based Model for Simulation of Color Vision Deficiency. *IEEE Transactions on Visualization and Computer Graphics*. **15**(6), 1291–1298. doi: 10.1109/TVCG.2009.113 Online version with supplements at http://www.inf.ufrgs.br/~oliveira/pubs_files/CVD_Simulation/CVD_Simulation.html.

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

cvd

Examples

```
# simulate color-vision deficiency by calling `simulate_cvd` with specified matrix
simulate_cvd(c("#005000", "blue", "#00BB00"), tritanomaly_cvd["6"][[1]])

# simulate color-vision deficiency by calling the shortcut high-level function
tritan(c("#005000", "blue", "#00BB00"), severity = 0.6)

# simulate color-vision deficiency by calling `simulate_cvd` with interpolated cvd matrix
simulate_cvd(c("#005000", "blue", "#00BB00"),
             interpolate_cvd_transform(tritanomaly_cvd, severity = 0.6))

# apply CVD directly on RGB matrix
RGB <- t(hex2RGB(rainbow(3))@coords*255)
deutan(RGB)
```

---

specplot *Color Spectrum Plot*

Description

Visualization of color palettes (given as hex codes) in HCL and/or RGB coordinates.

Usage

```
specplot(
  x,
  y = NULL,
  rgb = FALSE,
  hcl = TRUE,
  fix = TRUE,
  cex = 1,
  type = "l",
  lwd = 2 * cex,
```

```
  lty = 1,
  pch = NULL,
  mar = NULL,
  oma = NULL,
  main = NULL,
  legend = TRUE,
  palette = TRUE,
  plot = TRUE,
  ...
)
```

Arguments

x	character vector containing color hex codes.
y	optional second character vector containing further color hex codes, to be used for comparing two palettes (x vs. y).
rgb	logical or color specification. Should the RGB spectrum be visualized? Can also be a vector of three colors for the legend of R/G/B coordinates.
hcl	logical or color specification. Should the HCL spectrum be visualized? Can also be a vector of three colors for the legend of H/C/L coordinates.
fix	logical. Should the hues be fixed to be on a smooth(er) curve? For details see below.
cex	numeric. Character extension for figure axes and labels.
type, lwd, lty, pch	plotting parameters passed to lines for drawing the RGB and HCL coordinates, respectively. Can be vectors of length 3.
mar, oma	numeric or logical. Either numeric vectors of length 4 giving the (outer) margins or a logical indicating whether mar/oma should be set.
main	character. Main title of the plot.
legend	logical. Should legends for the coordinates be plotted?
palette	logical. Should the given palette x be plotted?
plot	logical. Should the RGB and/or HCL coordinates be plotted?
...	currently not used.

Details

The function specplot transforms a given color palette in hex codes into their HCL (polarLUV) and/or RGB (sRGB) coordinates. As the hues for low-chroma colors are not (or poorly) identified, by default a smoothing is applied to the hues (fix = TRUE). Also, to avoid jumps from 0 to 360 or vice versa, the hue coordinates are shifted suitably.

By default (plot = TRUE), the resulting HCL and optionally RGB coordinates are visualized by simple line plots along with the color palette x itself. The x-axis simply gives the ordering of the colors in the palette The y-axis depicts the following information: (1) Hue is drawn in red and coordinates are indicated on the axis on the right with range [0, 360] or (if necessary) [-360, 360]. (2) Chroma is drawn in green with coordinates on the left axis. The range [0, 100] is used unless

the palette necessitates higher chroma values. (3) Luminance is drawn in blue with coordinates on the left axis in the range [0, 100]. Luminance (and hence also chroma) is on the left axis because it is arguably most important for understanding the type of palette (qualitative vs. sequential vs. diverging). To facilitate reading the legend the reversed order Luminance / Chroma / Hue is used so that the legend labels are closer to the axis they pertain to.

For comparing two palettes, `specplot(x, y)` can be used which adds lines (dashed, by default) corresponding to the y palette HCL/RGB coordinates in the display.

Value

`specplot` invisibly returns a list with components

`HCL`	a matrix of HCL coordinates,
`RGB`	a matrix of sRGB coordinates,
`hex`	original color palette `x`.

Author(s)

Reto Stauffer, Achim Zeileis

References

Zeileis A, Hornik K, Murrell P (2009). Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis*, **53**, 3259–3270. doi: 10.1016/j.csda.2008.11.033 Preprint available from https://www.zeileis.org/papers/Zeileis+Hornik+Murrell-2009.pdf.

Stauffer R, Mayr GJ, Dabernig M, Zeileis A (2015). Somewhere over the Rainbow: How to Make Effective Use of Colors in Meteorological Visualizations. *Bulletin of the American Meteorological Society*, **96**(2), 203–216. doi: 10.1175/BAMSD1300155.1

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

See Also

`hcl_palettes`, `hclplot`

Examples

```
## spectrum of the (in)famous RGB rainbow palette (in both RGB and HCL)
specplot(rainbow(100), rgb = TRUE)

## spectrum of HCL-based palettes: qualitative/sequential/diverging
specplot(qualitative_hcl(100, "Set 2"))
specplot(sequential_hcl(100, "Blues 2"))
specplot(diverging_hcl(100, "Blue-Red"))

## return computed RGB and HCL coordinates
res <- specplot(rainbow(10), plot = FALSE)
print(res)
```

sRGB *Create sRGB Colors*

Description

This function creates colors of class sRGB; a subclass of the virtual color-class class.

Usage

```
sRGB(R, G, B, names)
```

Arguments

R, G, B	these arguments give the red, green and blue intensities of the colors (the values should lie between 0 and 1). The values can be provided in separate R, G and B vectors or in a three-column matrix passed as R.
names	A vector of names for the colors (by default the row names of R are used).

Details

This function creates colors in the standard sRGB color space (IEC standard 61966).

Value

An object of class sRGB which inherits from class color.

Author(s)

Ross Ihaka

See Also

RGB, HSV, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
# Create a random set of colors
set.seed(1)
sRGB(R = runif(20), G = runif(20), B = runif(20))
```

`swatchplot` *Palette Swatch Plot*

Description

Visualization of color palettes in columns of color swatches.

Usage

```
swatchplot(
  x,
  ...,
  nrow = 20,
  border = NULL,
  sborder = NULL,
  off = NULL,
  mar = NULL,
  line = NULL,
  cex = NULL,
  font = 1:2,
  cvd = FALSE
)
```

Arguments

`x`	character vector/matrix (or list of character vectors/matrices) containing color hex codes.
`...`	further (possibly named) character vectors/matrices with color hex codes.
`nrow`	integer specifying the maximal number of rows of swatches. (The actual number might be lower in order to balance the rows used in each column.)
`border`	color for border of individual color rectangles. By default `"lightgray"` for up to 9 colors, `"transparent"` otherwise.
`sborder`	color for border of the entire palette swatch. By default `"lightgray"` if `border` is `"transparent"` and `"lightgray"` otherwise (if `off = 0`).
`off`	numeric vector of length 2. Offset in horizontal and vertical direction (specified as a fraction of the rectangle for one color). By default, the horizontal offset is `0.3` for up to 5 colors and `0` otherwise, and the vertical offset is `0.1`.
`mar`	numeric vector of length 4, specifying the margins of column of color swatches.
`line`	numeric. Line in which the palette names (if any) are printed in the margin.
`cex, font`	numeric vectors of length 1 or 2. Specifications for the annotation text for the individual palettes and lists of palettes, respectively.
`cvd`	logical or character indicating whether color vision deficiencies should be emulated with `desaturate`, `deutan`, `protan`, `tritan`.

Details

The function swatchplot is a convenience function for displaying collections of palettes that can be specified as lists or matrices of character color specifications. Essentially, the function just calls rect but the value-added are the heuristics used for choosing default labels, margins, spacings, borders. These are selected to work well for hcl_palettes and might need further tweaking in future versions.

Value

swatchplot invisibly returns a matrix with colors and annotations.

References

Zeileis A, Fisher JC, Hornik K, Ihaka R, McWhite CD, Murrell P, Stauffer R, Wilke CO (2020). "colorspace: A Toolbox for Manipulating and Assessing Colors and Palettes." *Journal of Statistical Software*, **96**(1), 1–49. doi: 10.18637/jss.v096.i01

Examples

```
## swatches of several palette vectors
swatchplot(
  "Hue"       = sequential_hcl(5, h = c(0, 300), c = c(60, 60), l = 65),
  "Chroma"    = sequential_hcl(5, h = 0, c = c(100, 0), l = 65, rev = TRUE, power = 1),
 "Luminance" = sequential_hcl(5, h = 260, c = c(25, 25), l = c(25, 90), rev = TRUE, power = 1),
  off = 0
)

## swatches of named palette matrices
bprg <- c("Blues", "Purples", "Reds", "Greens")
swatchplot(
  "Single-hue"            = t(sapply(paste(bprg, 2), sequential_hcl, n = 7)),
  "Single-hue (advanced)" = t(sapply(paste(bprg, 3), sequential_hcl, n = 7)),
  "Multi-hue (advanced)"  = t(sapply(bprg,           sequential_hcl, n = 7)),
  nrow = 5
)

## swatches with color vision deficiency emulation
swatchplot(sequential_hcl(7, "Viridis"), cvd = TRUE)
swatchplot(
  "YlGnBu"  = sequential_hcl(7, "YlGnBu"),
  "Viridis" = sequential_hcl(7, "Viridis"),
  cvd = c("deutan", "desaturate")
)
```

USSouthPolygon *Polygon for County Map of US South States: Alabama, Georgia, and South Carolina*

Description

County polygons for Alabama, Georgia, and South Carolina plus an artifical variable used for coloring.

Usage

```
data("USSouthPolygon")
```

Format

A data frame with coordinates of the vertices of the county polygons (x, y) and an artificial variable z constructed for illustrating colored maps.

Source

Polygon data taken from **maps** package of Becker, Wilks, Brownrigg, and Minka (2012). Version 2.2-6. https://CRAN.R-project.org/package=maps

Examples

```
## generate color palette
pal <- diverging_hcl(9)
n <- length(pal)

## draw shaded polygons
plot(0, 0, type = "n", xlab = "", ylab = "", xaxt = "n", yaxt = "n", bty = "n",
  xlim = c(-88.5, -78.6), ylim = c(30.2, 35.2), asp = 1)
polygon(USSouthPolygon, col = pal[cut(na.omit(USSouthPolygon$z), breaks = 0:n/n)])
```

---

whitepoint *Access or Modify the Whitepoint*

Description

This function can be used to control the single global whitepoint that affects all color conversions within the package (that require a whitepoint, i.e., go through XYZ).

Usage

```
whitepoint(white, ...)
```

Arguments

white, ...	Either missing (to query the whitepoint) or NULL or a specification of the XYZ coordinates of the whitepoint (to set the whitepoint, see examples). NULL corresponds to CIE D65 with XYZ coordinates 95.047, 100.000, 108.883.

Value

`whitepoint` returns an XYZ color object for the whitepoint (invisibly in case a new whitepoint was set).

See Also

XYZ and color-class.

Examples

```
# query current whitepoint (D65 by default)
whitepoint()

# Illuminant E
whitepoint(XYZ(100, 100, 100))

# equivalently
whitepoint(100, 100, 100)
whitepoint(c(100, 100, 100))
whitepoint(cbind(100, 100, 100))

whitepoint()

## reset
whitepoint(NULL)
whitepoint()
```

---

writehex *Write Hexadecimal Color Descriptions*

Description

Given a color object, this function writes a file containing the hexadecimal representation of the colors in the object.

Usage

```
writehex(x, file = "")
```

Arguments

`x`	a color object.
`file`	the name of the file to be written.

Details

This function converts the given color object to RGB and then writes hexadecimal strings (of the form `#RRGGBB`) representing the colors to the specified file.

Value

The name of the file is returned as the value of the function.

Author(s)

Ross Ihaka

See Also

readhex, readRGB, hex2RGB, RGB, HSV, XYZ, LAB, polarLAB, LUV, polarLUV.

Examples

```
set.seed(1)
x <- sRGB(runif(10), runif(10), runif(10))
## IGNORE_RDIFF_BEGIN
writehex(x, file.path(tempdir(), "random.txt"))
## IGNORE_RDIFF_END
```

XYZ *Create XYZ Colors*

Description

This function creates colors of class XYZ; a subclass of the virtual color-class class.

Usage

```
XYZ(X, Y, Z, names)
```

Arguments

X, Y, Z	these arguments give the X, Y and Z coordinates of the colors. The values can be provided in separate X, Y and Z vectors or in a three-column matrix passed as X.
names	A vector of names for the colors (by default the row names of X are used).

Details

The X, Y and Z values are the levels of the CIE primaries. These are scaled so that the luminance of the display white-point is 100. The white-point is taken to be D65, which means that its coordinates are 95.047, 100.000, 108.883.

Value

An object of class XYZ which inherits from class color.

Author(s)

Ross Ihaka

See Also

RGB, HSV, LAB, polarLAB, LUV, polarLUV.

Examples

```
## Generate white in XYZ space
XYZ(95.047, 100.000, 108.883)
```

Index