Package 'colorpatch'

October 12, 2022

Type Package

Title Optimized Rendering of Fold Changes and Confidence Values

Description
Shows color patches for encoding fold changes (e.g. log ratios) together with confidence values within a single diagram. This is especially useful for rendering gene expression data as well as other types of differential experiments. In addition to different rendering methods (ggplot extensions)
functionality for perceptually optimizing color palettes are provided.
Furthermore the package provides extension methods of the colorspace color-class in order to simplify the work with palettes (a.o. length, as.list, and append are supported).

Version 0.1.2

Date 2017-06-08

License Artistic-2.0

Depends R (>= 3.0.0)

Imports ggplot2, colorspace, methods, grid, gridExtra, stats, TSP, utils

Suggests plotly, knitr, rmarkdown, testthat

RoxygenNote 6.0.1

LazyData FALSE

VignetteBuilder knitr

URL http://sysbio.uni-ulm.de/?Software:colorpatch

Collate 'colorpatch.R' 'colorpatch_methods.R' 'colorpatch_ggplot.R' 'colorpatch_impl.R' 'data.R'

NeedsCompilation no

Author Andre Mueller [aut, cre], Hans Kestler [aut]

Maintainer Andre Mueller <andre@kiwisound.de>

Repository CRAN

Date/Publication 2017-06-10 06:22:56 UTC

R topics documented:

colorpatch-package *A small introduction to the colorpatch package.*

Description

The colorpatch package provides functions for plotting **color patch grids** rendering the two channels fold change and confidence value within a single diagram. This is especially useful for analyzing gene expression data as well as other types of "change" data such as gains/losses in stock exchange or analyzing the agricultural output.

Details

The packages consists of:

- ggplot extensions for visualizing color patch grids `colorpatch::stat_colorpatch()` and `colorpatch::stat_bicolor()`
- Functionality for rearranging data for a better readable map `colorpatch::OrderData()`
- Perceptual optimization functions for sub-sampling non-uniform bicolored palettes `colorpatch::OptimizeBiColor()`

For more details see the vignette

Author(s)

Maintainer: Andre Mueller <andre@kiwisound.de>

Authors:

- Hans Kestler <hans.kestler@uni-ulm.de>

See Also

Useful links:

- http://sysbio.uni-ulm.de/?Software:colorpatch

Examples

```
vignette("introduction", package = "colorpatch")
```

`append,color-method` *Appends two palettes to form a single palette.*

Description

Applies to the colorspace::color class.

Usage

```
## S4 method for signature 'color'
append(x, values, after = length(x))
```

Arguments

`x`	the color palette to be modified.
`values`	another color palette to be appended
`after`	currently unimplemented.

`apply.color` *Applies a function to each entry of a colorspace::color palette.*

Description

Applies a function to each entry of a colorspace::color palette.

Usage

```
apply.color(X, FUN, ...)
```

Arguments

`X`	the color palette
`FUN`	the function to be applied
`...`	extra arguments to FUN

Value

a list of each result of FUN applied to each entry in X

as *Transforms palette to list of single colors.*

Description

Applies to the colorspace::color class.

Examples

```
data("OptimGreenRedLAB")
as(OptimGreenRedLAB, "list")
```

as.list *Creates a list with single colors from a palette.*

Description

Applies to the colorspace::color class.

Usage

```
as.list(x, ...)

## S4 method for signature 'color'
as.list(x, ...)
```

Arguments

x	color object to be coerced to a list
...	ignored for this class

Examples

```
data("OptimGreenRedLAB")
as.list(OptimGreenRedLAB)
```

ColorDistance *Computes the perceptional distance between two neighboring colors*

Description

Computes the perceptional distance between two neighboring colors

Usage

```
ColorDistance(pal, color.space = "LAB")
```

Arguments

pal	the color palette
color.space	color space in which the distance shall be computed (default "LAB")

Value

a vector of distances

Examples

```
data("OptimGreenRedLAB")
dd <- ColorDistance(OptimGreenRedLAB)
```

ColorPatchColorFun *Creates a color function mapping (ratio, conf) tuples to a single color*

Description

Creates a color function mapping (ratio, conf) tuples to a single color

Usage

```
ColorPatchColorFun(palette = "OptimGreenRedLAB")
```

Arguments

palette	name of the palette (see data()) - defaults to "OptimGreenRedLAB"

Value

A function mapping (ratio, conf) to a color.

Examples

```
fn <- ColorPatchColorFun("OptimBlueYelloLAB")
```

ColorPatchSizeFun *Creates a size function mapping (ratio, conf) to a single color*

Description

Creates a size function mapping (ratio, conf) to a single color

Usage

```
ColorPatchSizeFun(type = "linear")
```

Arguments

type	defaults to "linear"

Value

A function mapping (ratio, conf) to a size.

ColorRgbFun *Creates a color mapping function*

Description

Creates a color mapping function

Usage

```
ColorRgbFun(pal, xmin = -1, xmax = 1, coerce.fun = colorspace::hex)
```

Arguments

pal	the color palette
xmin	minimum value to be mapped to the first entry of the palette
xmax	maximum value to be mapped to the last entry of the palette
coerce.fun	the color coercing function (e.g. for ggplot2 colorspace::hex() is recommended)

Value

a function mapping a value to a color

Examples

```
data("OptimGreenRedLAB")
fn <- ColorRgbFun(OptimGreenRedLAB)
```

ComputeSymmetry *Computes the symmetry of a given bi-variate color palette*

Description

Computes the symmetry of a given bi-variate color palette

Usage

```
ComputeSymmetry(pal, color.space = "LAB")
```

Arguments

pal	A two-sided input palette colorspace::color
color.space	Color space where the distances shall be computed (default "LAB")

Value

a data frame with index, side (pos/neg) and distance

Examples

```
data("OptimGreenRedLAB")
df <- ComputeSymmetry(OptimGreenRedLAB)
print(df)
```

CreateClusteredData *Creates clustered random data*

Description

Creates clustered random data

Usage

```
CreateClusteredData(nrow = 30, ncol = 12, nrow.clusters = 2,
  ncol.clusters = 2, alpha = 4)
```

Arguments

nrow	Number of rows (default: 30)
ncol	Number of columns (default: 12)
nrow.clusters	Number of row cluster
ncol.clusters	Number of column clusters (default: 2)
alpha	Scaling facor (default: 4)

Value

A data set with $ratio and $conf values

`CreateExampleData` *Creates demonstration data of the colorpatch package*

Description

Creates demonstration data of the colorpatch package

Usage

```
CreateExampleData(nrow = 30, ncol = 12)
```

Arguments

`nrow`	number of rows (default 30)
`ncol`	number of columns (default 12)

Value

the data set

Examples

```
library(ggplot2)
library(colorpatch)
dat <- CreateExampleData()
df <- ToDataFrame(dat)
p <- ggplot(df, aes(x = x, y = y, ratio = ratio, conf = conf))
p <- p + theme_colorpatch() + coord_fixed(ratio = 1)  + stat_colorpatch()
plot(p)
```

`DistColor` *Computes the distance of to colors within a certain colorspace*

Description

Computes the distance of to colors within a certain colorspace

Usage

```
DistColor(x, y, color.space = "LAB")
```

Arguments

x	First color to be compared
y	Second color to be compared
color.space	Defaults to "LAB" (can be anything within the colorspace package) see colorspace::color

Value

L2 distance of the two colors within the given coordinate space

See Also

colorspace::color, DistColorFun()

Examples

```
library(colorspace)
library(colorpatch)
DistColor(sRGB(0.1,0.5,0), sRGB(0.2,0.7,1.0), "LUV")
```

DistColorFun *Creates a color distance function*

Description

Creates a color distance function

Usage

```
DistColorFun(color.space = "LAB")
```

Arguments

color.space	Color space to be used (see colorspace::color)

Value

A function mapping two color values of a color class colorspace::color to a numeric value.

Examples

```
library(colorspace)
library(colorpatch)
fn <- DistColorFun("LUV")
a <- sRGB(1,0,0)
b <- sRGB(0.8,0.1,0)
my.distance <- fn(a,b)
```

FindUniformSequence *Finds a uniform color sequence within a non-uniform palette by sub-sampling that palette*

Description

Finds a uniform color sequence within a non-uniform palette by subsampling that palette

Usage

```
FindUniformSequence(P, n.out, reverse = FALSE, delta = NULL,
  col.dist.fun = DistColorFun("LAB"))
```

Arguments

P	input color palette (must be a class derived from colorspace::color)
n.out	number of output colors (must be less than length(P))
reverse	shall the searching be performed from the end of the palette to the beginning
delta	the perceptual difference to be achieved between two adjecent colors
col.dist.fun	function mapping two colors to a numeric distance

Value

a optimized palette (sub-set of P)

GeneratePalettes *Creates color palettes and saves them as files*

Description

Creates color palettes and saves them as files

Usage

```
GeneratePalettes(col.dist.fun = DistColorFun("LAB"), ...)
```

Arguments

col.dist.fun	Color distance function.
...	Additional arguments forwarded to colorpatch::OptimizeBiColor().

Value

Nothing - this function is used for its side effects (creating files in data).

GreenRedRGB — *Standard RGB Green/Red two-sided color scale.*

Description

A two-sided color scale left side: green, center: black, right side: red.

Usage

```
GreenRedRGB
```

Format

An object of class colorspace::color.

HsvColorFun — *Creates a color function mapping ratio/conf values to a HSV colorspace*

Description

Creates a color function mapping ratio/conf values to a HSV colorspace

Usage

```
HsvColorFun(coerce.fn = colorspace::hex, hue.offset = 60, hue.scale = -60,
  saturation = 1)
```

Arguments

coerce.fn	coerces each HSV color with this function (defaults colorspace::hex())
hue.offset	hue offset (defaults to 60)
hue.scale	hue scale (defaults to 60)
saturation	HSV saturation (defaults to 1)

Value

a color mapping function (ratio,conf) -> color

`HsvSizeFun` *Creates a size function mapping ratio/conf to a patch size for bicolorings*

Description

Creates a size function mapping ratio/conf to a patch size for bicolorings

Usage

```
HsvSizeFun()
```

Value

a size mapping function (ratio,conf) -> size

`InterpolateColorFun` *Linear interpolation within a colorspace::color palette*

Description

This function can be used together with ggplot2 for mapping values onto colorspace::color palettes. The color is then coerced with `coerce.fun`.

Usage

```
InterpolateColorFun(pal, xmin = -1, xmax = +1,
  coerce.fun = colorspace::hex)
```

Arguments

`pal`	The input palette (must be of class colorspace::color)
`xmin`	minimum of the numeric range to be mapped onto `pal`
`xmax`	maximum of the numeric range to be mapped onto `pal`
`coerce.fun`	each color will be coerced by this function (defaults to `colorspace::hex()`)

Value

A function mapping a numeric value `value` onto a color value.

Examples

```
library(colorspace)
library(colorpatch)
data("OptimGreenRedLAB")
fn <- InterpolateColorFun(OptimGreenRedLAB)
cols <- fn(seq(-1, 1, by = 0.1))
specplot(cols)
```

length,color-method *Returns the length of a palette (the number of entries).*

Description

Applies to the colorspace::color class.

Usage

```
## S4 method for signature 'color'
length(x)
```

Arguments

x	an color object

LinColorSpace *Creates a linear color space between two colors*

Description

Creates a linear color space between two colors

Usage

```
LinColorSpace(color1, color2, n.out)
```

Arguments

color1	the first color (must be of the class colorspace::color)
color2	the second color (must be of the class colorspace::color)
n.out	number of output colors

Value

a palette

Examples

```
library(colorspace)
library(colorpatch)
pal <- LinColorSpace(sRGB(0,1,0), sRGB(0,0.1,0), 32)
pal <- append(pal, sRGB(0,0,0))
pal <- append(pal, LinColorSpace(sRGB(0.1,0,0), sRGB(1,0,0), 32))
PlotUniformity(pal)
print(pal)
```

OptimBlueYellowLAB *Optimum RGB Blue/Yellow two-sided color scale in LAB color space.*

Description

A two-sided color scale left side: blue, center: black, right side: yellow.

Usage

```
OptimBlueYellowLAB
```

Format

An object of class colorspace::color.

OptimGreenRedLAB *Optimum RGB Green/Red two-sided color scale in LAB color space.*

Description

A two-sided color scale left side: green, center: black, right side: red.

Usage

```
OptimGreenRedLAB
```

Format

An object of class colorspace::color.

OptimizeBiColor *Optimizes a bicolor palette*

Description

Optimizes a bicolor palette

Usage

```
OptimizeBiColor(neg.col.min = colorspace::sRGB(0, 0.01, 0),
  neg.col.max = colorspace::sRGB(0, 1, 0),
  pos.col.min = colorspace::sRGB(0.01, 0, 0),
  pos.col.max = colorspace::sRGB(1, 0, 0), center.col = colorspace::sRGB(0,
  0, 0), n.out = 64, oversampling = 128,
  col.dist.fun = DistColorFun("LAB"), reverse = FALSE)
```

Arguments

neg.col.min	color representing the negative mininum value
neg.col.max	color representing the negative maximum value
pos.col.min	color for the positive minimum value
pos.col.max	color representing the positive maximum value
center.col	center color which maps to 0 (default: black)
n.out	size of each half-palette
oversampling	the oversampling rate
col.dist.fun	color distance function (default: DistColorFun("LAB")) for optimizing the palette
reverse	shall the palette be searched starting from the minimum color to the maximum (reverse=FALSE) or vice versa - defaults to FALSE

Value

bicolor palette

Examples

```
pal <- OptimizeBiColor(n.out = 8, oversampling = 32)
PlotUniformity(pal)
```

OrderData *Orders rows and columns of data.*

Description

Orders rows and columns of data.

Usage

```
OrderData(dat, orderFn = OrderDataHclust, distFn = stats::dist)
```

Arguments

dat	Ratio data
orderFn	Ordering method (default: OrderDataHclust)
distFn	Distance function (Idefault stats::dist)

Value

ordered data

`OrderDataHclust` *Orders rows and column distances with* `stats::hclust()`

Description

Orders rows and column distances with `stats::hclust()`

Usage

```
OrderDataHclust(row.dist, col.dist, ...)
```

Arguments

`row.dist`	row distances
`col.dist`	column distances
`...`	optional parameters forwarded to the `stats::hclust()` function

Value

a list with irow and icol containing the orders of rows and columns

`OrderDataTSP` *Orders rows and column distances with traveling salesman ordering TSP*

Description

Orders rows and column distances with traveling salesman ordering TSP

Usage

```
OrderDataTSP(row.dist, col.dist, ...)
```

Arguments

`row.dist`	row distances
`col.dist`	column distances
`...`	optional parameters fed to the `TSP::solve_TSP()` function

Value

a list with irow and icol containing the orders of rows and columns

OrderWithTSP *Orders a data set given a distance matrix with TSP*

Description

Orders a data set given a distance matrix with TSP

Usage

```
OrderWithTSP(dist, ...)
```

Arguments

dist	distance object or distance matrix
...	extra arguments fed to TSP::solve_TSP()

Value

a path (vector of integers)

PlotSymmetry *Plots the symmetry of a bivariate color scale*

Description

Plots the symmetry of a bivariate color scale

Usage

```
PlotSymmetry(pal, color.space = "LAB")
```

Arguments

pal	A two-sided input palette colorspace::color
color.space	Color space where the distances shall be computed (default "LAB")

Value

a ggplot object

Examples

```
data("OptimGreenRedLAB")
PlotSymmetry(OptimGreenRedLAB)
```

PlotUniformity *Plots the uniformity of a color palette*

Description

Plots the uniformity of a color palette

Usage

```
PlotUniformity(pal, color.space = "LAB")
```

Arguments

pal	A colorspace palette
color.space	the color space (see colorspace::color)

Value

a ggplot instance

Examples

```
data("OptimGreenRedLAB")
p <- PlotUniformity(OptimGreenRedLAB)
plot(p)
```

ReadArraySRGB *Reads a sRGB color table as CSV file*

Description

Reads a sRGB color table as CSV file

Usage

```
ReadArraySRGB(file.name)
```

Arguments

file.name	the color file

Value

a colorspace palette

StatColorPatch *A ggplot2::ggproto class for showing color patches.*

Description

A ggplot2::ggproto class for showing color patches.

Usage

```
StatColorPatch
```

Format

An object of class `StatColorPatch` (inherits from `Stat`, `ggproto`) of length 4.

stat_bicolor *Plots a ratio/confidence plot using a bivariate colormap*

Description

Plots a ratio/confidence plot using a bivariate colormap

Usage

```
stat_bicolor(mapping = NULL, data = NULL, geom = "tile",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, color.fun = HsvColorFun(), size.fun = HsvSizeFun(),
  ...)
```

Arguments

`mapping`	Set of aesthetic mappings created by `ggplot2::aes()`. If specified and `inherit.aes = TRUE` (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
`data`	The data to be displayed in this layer.
`geom`	Defaults to `tile`.
`position`	Position adjustment, either as a string, or the result of a call to a position adjustment function.
`na.rm`	If `FALSE`, the default, missing values are removed with a warning. If `TRUE`, missing values are silently removed.
`show.legend`	logical. Should this layer be included in the legends? `NA`, the default, includes if any aesthetics are mapped. `FALSE` never includes, and `TRUE` always includes.

inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
color.fun	Color function mapping a (ratio,conf) pair to a color (defaults to colorpatch::HsvColorFun()).
size.fun	Size function mapping a (ratio,conf) pair to a rectangle size (defaults to colorpatch::HsvSizeFun() returning constantly 1).
...	further arguments given to the StatColorPatch() function

Value

a ggplot statistics layer for showing bicolored maps

Examples

```
library(ggplot2)
library(colorpatch)
dat <- CreateExampleData()
df <- ToDataFrame(dat)
p <- ggplot(df) + theme_colorpatch() + stat_bicolor(aes(ratio=ratio,conf=conf,x=x,y=y))
```

---

stat_colorpatch *A stat function for the use with ggplot2*

Description

A stat function for the use with ggplot2

Usage

```
stat_colorpatch(mapping = NULL, data = NULL, geom = "tile",
  position = "identity", na.rm = FALSE, show.legend = NA,
  inherit.aes = TRUE, color.fun = ColorPatchColorFun(),
  size.fun = ColorPatchSizeFun(), ...)
```

Arguments

mapping	Set of aesthetic mappings created by ggplot2::aes(). If specified and inherit.aes = TRUE (the default), it is combined with the default mapping at the top level of the plot. You must supply mapping if there is no plot mapping.
data	The data to be displayed in this layer.
geom	Defaults to tile.
position	Position adjustment, either as a string, or the result of a call to a position adjustment function.
na.rm	If FALSE, the default, missing values are removed with a warning. If TRUE, missing values are silently removed.

show.legend	logical. Should this layer be included in the legends? NA, the default, includes if any aesthetics are mapped. FALSE never includes, and TRUE always includes.
inherit.aes	If FALSE, overrides the default aesthetics, rather than combining with them. This is most useful for helper functions that define both data and aesthetics and shouldn't inherit behaviour from the default plot specification, e.g. borders.
color.fun	Color function mapping a (ratio,conf) pair to a color (defaults to colorpatch::ColorPatchColorFun()).
size.fun	Size function mapping a (ratio,conf) pair to a rectangle size (defaults to colorpatch::ColorPatchSizeFu returning constantly 1).
...	Further arguments given to the colorpatch::StatColorPatch ggproto object. Here thresh.ratio, thresh.conf are the most important parameters.

Value

a ggplot statistics layer for showing color patches

---

theme_colorpatch *A ggplot2 theme for rendering colorpatches (black background)*

Description

A ggplot2 theme for rendering colorpatches (black background)

Usage

```
theme_colorpatch(fill = "black", plot.background = fill)
```

Arguments

fill	background fill color (default: "black")
plot.background	background fill color (default: "black")

Value

a theme function for showing color patches

Examples

```
library(ggplot2)
library(colorpatch)
dat <- CreateExampleData()
df <- ToDataFrame(dat)
p <- ggplot(df) + theme_colorpatch() + stat_colorpatch(aes(ratio=ratio,conf=conf,x=x,y=y))
```

`ToDataFrame` *Transforms a ratio/conf data set to a ggplot dataframe*

Description

Transforms a ratio/conf data set to a ggplot dataframe

Usage

```
ToDataFrame(dat)
```

Arguments

`dat`	must be a list with two matrices `ratio` and `conf`

Value

a data frame

Index