Package 'Polychrome'

May 6, 2019

Title Qualitative Palettes with Many Colors

Version 1.2.2

Date 2019-05-01

Author Kevin R. Coombes, Guy Brock

Description Tools for creating, viewing, and assessing qualitative palettes with many (20-30 or more) colors.

Maintainer Kevin R. Coombes <krc@silicovore.com>

Depends R (>= 3.0)

Imports colorspace, scatterplot3d, methods, graphics, grDevices, stats, utils

Suggests RColorBrewer, knitr, rmarkdown

License Apache License (== 2.0)

LazyLoad yes

LazyData no

URL http://oompa.r-forge.r-project.org/

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2019-05-06 16:20:03 UTC

R topics documented:

`alphabet` *A 26-Color Palette*

Description

A palette composed of 26 distinctive colors with names corresponding to letters of the alphabet.

Usage

```
data(alphabet)
```

Format

A character string of length 26.

Details

A character vector containing hexadecimal color representations of 26 distinctive colors that are well separated in the CIE L*u*v* color space.

Source

The color palette was generated using the `createPalette` function with three seed colors: ebony ("#474747"), iron ("#E2E2E2"), and red ("#F70000"). The colors were then manually assigned names begining with different letters of the English alphabet.

See Also

`createPalette`

Examples

```
data(alphabet)
alphabet
```

colorDeficit *Converting Colors to Illustrate Color Deficient Vision*

Description

Function to convert any palette to one that illustrates how it would appear to a person with a color deficit.

Usage

```
colorDeficit(rgb, target = c("deuteranope", "protanope", "tritanope"))
```

Arguments

rgb	A color palette. Accepts hexademical representations, sRGB class objects from the colorspace package, or three-column sRGB matrices.
target	The kind of color deficit to simulate.

Details

This function converts normal-vision color palettes into simulations that represent what is likely to be seen with one of the three kinds of color deficits. Deuteranopes are red-blind, which is the most common form of color deficit leading to an inability to distinguish red and green. Protanopes are green-blind; this is the second most common form of color-blindness and also leads to an inability to distinguish red and green. Tritanopes are blue blind; this is the rarest form of color blindness and leads to an inability to distinguish blue and yellow.

Value

Returns a color palette in the same form as its input argument.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References

[1] http://www.vischeck.com/

[2] Brettel H, Vienot F, Mollon JD. Computerized simulation of color appearance for dichromats. J Opt Soc Am A Opt Image Sci Vis. 1997 Oct;14(10):2647-55. PubMed PMID: 9316278.

[3] Vienot F, Brettel H, Ott L, Ben M\'Barek A, Mollon JD. What do colour-blind people see? Nature. 1995 Jul 13;376(6536):127-8. PubMed PMID: 7603561.

See Also

color-class

Examples

```
alfa <- alphabet.colors(26)
def <- colorDeficit(alfa)
swatch(def)
```

`colorsafe` *A 10-Color Palette Distinguishable By COlor-Deficit Individuals*

Description

A palette composed of 10 distinctive colors, selected to be distinguishable by a mojority of individuals who have some form mof color deficient vision.

Usage

```
data(colorsafe)
```

Format

A character string of length 10.

Details

A character vector containing hexadecimal color representations of 10 distinctive colors that are well separated in the CIE L*u*v* color space, chosen to be distinguishable by people with color deficient vision.

Source

Details of how te paklette was constructed can be found in the "color-deficit" vignette.

See Also

`createPalette`

Examples

```
data(colorsafe)
colorsafe
## Not run: vignette("color-deficits")
```

createPalette *Creating New Color Palettes*

Description

Tool to create new palettes that are well separated in CIE L*u*v* color space.

Usage

```
createPalette(N, seedcolors, prefix = "NC", range = c(30, 90),
              target = c("normal", "protanope", "deuteranope", "tritanope"),
              M = 50000)
```

Arguments

N	An integer, the size of the palette to create.
seedcolors	A character vector containing the hexadecimal representations of one or more colors.
prefix	A character string to be used as a prefix to numeric names of the colors.
range	A numeric vector limitng the range of allowed luminance values.
target	A character string indicating the kind of color vision for which the palette is intended.
M	An integer; the number of random colors to generate while creating palettes.

Details

Carter and Carter showed that "perceptual distinguishability" of colors was related to their Euclidean distance in the L*u*v* color space coordinates, as defined by the International Commisision on Illumination (CIE). The createPalette function implements a greedy algorithm to find colors that are well-spread-out in L*u*v* space. The algorithm begins by generating a random set of 50,000 colors; these colors are restricted to those whose luminance lies between 30 and 90. Then, given one or more starting colors, the algorithm finds the random color that maximizes the distance to the closest existing color point. This process continues until N colors have been selected.

Value

A character string containing the hexadecimal representations of N colors that are well spread out in CIE L*u*v* color space.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References

Carter RC, Carter EC. High-contrast sets of colors. Applied Optics, 1982; 21(16):2936–9.

See Also

Color Palettes, colorDeficit.

Examples

```
seed <- c("#ff0000", "#00ff00", "#0000ff")
mycolors <- createPalette(15, seed, prefix="mine")
swatch(mycolors)
```

Dark24 *Light and Dark 24-Color Palettes*

Description

Two palettes, each composed of 24 distinctive colors, optimized for either a light background (Dark24) or a dark background (Light24).

Usage

```
data(Dark24)
data(Light24)
```

Format

A character vector of length 24.

Details

A character vector containing hexadecimal color representations of 24 distinctive colors that are well separated in the CIE L*u*v* color space.

Source

Both color palettes were generated using the createPalette function. In addition to specifing seed colors, the luminance range was restricted to produce either only light colors or only dark colors.

See Also

createPalette

Examples

```
data(Dark24)
Dark24
data(Light24)
swatch(Light24)
```

`distances` *Visualizing Color Palettes*

Description

Functions that provide visualization of palettes to help determine appropriate contexts where thay can be used.

Usage

```
computeDistances(colorset)
plotDistances(colorset, main=deparse(substitute(colorset)), pch=16, ...)
```

Arguments

`colorset`	a character vector containing hexadecimal color values.
`main`	a character string, the main title for a plot
`pch`	Plotting character to use.
`...`	additional graphical parameters.

Details

Carter and Carter established the fact that, for two colors to be reliably distinguished, the Euclidean distance between their representations in CIE L*u*v* color space should be at least 40 units. The `computeDistances` function reorders the colors by maximal separation in L*u*v* space, and computes the minimum distance of the next color to all the preceeding colors. The `plotDistances` function computes distances and immediately plots the result.

Value

The `plotDistances` function returns a list with two vector components: the `colors` in sorted order, and the minimum `distances` from each color to the set of preceeding colors. The `computeDistances` function returns the vector of minimum distances.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References

Carter RC, Carter EC. High-contrast sets of colors. Applied Optics, 1982; 21(16):2936–9.

See Also

`palette.viewers`

Examples

```
data(alphabet)
plotDistances(alphabet)
luvd <- computeDistances(alphabet)
```

---

getLUV *Get the L*u*v* coordinates of the colors*

Description

Given a character vector of colors in any format acceptable to R, this function computes their coordinates in L*u*v* color space.

Usage

```
getLUV(colorset)
```

Arguments

colorset	a vector containing color values in any format recognized by R.

Details

The real point of this function is to allow users to plot the colors in three-dimensional space using the rgl package. Because rgl depends on an external installation of XQuartz on Macintosh computers (and because we have found that some students in courses that we teach are apparently unable to install XQuartz, especially if they are using institutional computers without adminstrative privileges), Polychrome no longer imports or depends upon the rgl package, instead relying on scatterplot3d for its 3D plots. The example below shows how to plot a color set using rgl.

Value

Returns a list containing two components: (1) a three-column matrix named coords containing the luminance (L) and hue coordinates (U, V) of each color provided in the input colorset; and (2) a character vector named cset containng a hexadecimal representation of the colorset.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

Examples

```
data(alphabet)
luv <- getLUV(alphabet)
scatterplot3d::scatterplot3d(luv$coords, color = luv$cset,
    pch = 16, cex.symbol = 3)
## Not run:
library(rgl)
```

```
x <- luv$coords
cset <- luv$cset
open3d(windowRect=c(40, 40, 840, 840))
plot3d(x, main="Alphabet Colors")
spheres3d(x, radius=10, col=cset, shininess=100)

## End(Not run)
```

`glasbey` *The 32-color Glasbye palette*

Description

A palette composed of 32 distinct colors.

Usage

```
data(glasbey)
```

Format

A character string of length 32.

Details

A character vector containing hexadecimal color representations of 32 distinctive colors that are well separated in the CIE L*u*v* color space.

Source

The color palette was created, using standard tools in the `colorspace` package from a manually transcribed matrix of RGB values copied from the paper by Glasbey and colleagues.

References

Glasbey CA, van der Heijden GWAM, Toh VFK, Gray AJ (2007). Colour Displays for Categorical Images. Color Research and Application, 32, 304-9.

Examples

```
data(glasbey)
head(glasbey)
```

`invertColors` *Inverting the Plot Device Color Scheme*

Description

Function to convert the default plot color scheme to white-on-black.

Usage

```
invertColors(...)
```

Arguments

`...`	Other graphical parameters to be given to `par`.

Details

This function changes the default color scheme of the current graphics device to white on black. Note that since `invertColors` resets the `bg` parameter, you should avoid passing in a new default value for the `col` parameter.

Value

It returns the original color scheme, which can be passed to the `par` command to restore the original values.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

`par`

Examples

```
opar <- invertColors()
plot(1:3, 4:6, pch=16)
par(opar)
```

iscc *Color Names From the Inter-Society Color Council (ISCC)*

Description

A data frame mapping hex codes for 267 colors to their official ISCC-NBS names.

Usage

```
data(iscc)
```

Format

A data frame with three columns and 267 rows.

Details

This data set contains short names, long names, and hex codes for the 267 official color namkes defineed by the ISCC. Data was obtained from the Texas Precancel CLub and reformatted to be used conveniently in R.

Source

Our main source was the no-longer-extant web site of the Texas Precancel Club (http://tx4.us/nbs-iscc.htm).

References

See the Inter-Society Color Council web site (http://www.iscc.org/); the Wikipedia article on the ISCC-NBS system of color designation (https://en.wikipedia.org/wiki/ISCC%E2%80%93NBS_system.

See Also

isccNames

Examples

```
data(iscc)
head(iscc)
```

`isccNames` *Standard Names for Colors*

Description

The Inter-Society Color Council, in cooperation with the United States National Bureau of Standards, developed a list of 267 standardized color names. Many software tools (including R) also use a (non-standardized) list of color names derived from the original X11 list on early UNIX systems. We provide tools to convert hexadecimal colors to both sets of names.

Usage

```
isccNames(colorset)
colorNames(colorset)
```

Arguments

`colorset`	A character vector containing hexadecimal representations of colors.

Details

Each of the ISCC-NBS 267 standard color names is represented by the centroid of a region of CIE L*u*v* color space, all of whose points should be given the same name. Each of the color names listed by the `colors` function has an associated RGB color that can also be converted to L*u*v* space. These functions take colors represented in the common hexadecimal notation, maps them into L*u*v* color space, and assigns the name of the nearest ISCC centroid or UNIX/X11/R color.

Value

A character string containing the standard color name nearest (in CIE L*u*v* color space) to each input color.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References

Kelly KL. Twenty-Two Colors of Maximum Contrast. Color Eng., 1965; 3:26–7.

Also see the Inter-Society Color Council web site (`http://www.iscc.org/`).

See Also

`iscc`, `colors`.

Examples

```
data(alphabet)
isccNames(alphabet)
colorNames(alphabet)
```

---

palette.viewers *Visualizing Color Palettes*

Description

Functions that provide visualization of palettes to help determine appropriate contexts where thay can be used.

Usage

```
rancurves(colorset, ...)
ranpoints(colorset, N=10, ...)
swatch(colorset, main=deparse(substitute(colorset)))
swatchHue(colorset, main=paste(deparse(substitute(colorset)),
                           ", by Hue", sep=""))
swatchLuminance(colorset, main=paste(deparse(substitute(colorset)),
                           ", by Luminance", sep=""))
ranswatch(colorset, main=deparse(substitute(colorset)))
uvscatter(colorset, main=deparse(substitute(colorset)), ...)
luminance(colorset, main=deparse(substitute(colorset)), ...)
plothc(colorset, main=deparse(substitute(colorset)), ...)
plotpc(colorset, main=deparse(substitute(colorset)), ...)
p3d(colorset, main=deparse(substitute(colorset)), ...)
```

Arguments

colorset	a character vector containing hexadecimal color values.
main	a character string, the main title for a plot
N	an integer; the number of points to plot in each color.
...	additional graphical parameters.

Details

Different palettes are useful in different contexts. For example, high luminance colors may work well in barplots but provide low contrast when used to color points in scatter plots. The best way to decide if a palette is right for any particular application is probably to create a sample plot using the palette. The functions described here provide sample plots that display colors.

The function rancurves produces a set of sine curves with different phases and amplitudes, with each curve shown in a different color. The function ranpts produces a scatter plot showing N clustered points in each of the palette colors.

There are four functions that use barplots to display the palette. The simplest one, `swatch`, simply produces one bar of height one for each color, in the order that they are listed in the palette. The next two, `swatchHue` and `swatchLuminance`, first sort the palette (by hue or by luminance, respectively), before producing the barplot. The goal of these functions is to make sure that similar colors can be distinguished by placing them close together. The final function, `ranswatch`, randomly sorts the colors, to help decide if similar colors are identifiable when they are relatively far apart.

The `p3d` function plots the palette colors as spheres in three-dimensional CIE L*u*v* color space. It has been shown that perceptual distance is closely related to Euclidean distance in L*u*v* space. The `uvscatter` function produces a scatter plot of the palette colors using their projected u-v coordinates. The `luminance` function sorts the colors by luminance and produces a scatter plot showing the luminance.

The `plothc` function performs hierarchical clustering on the colors (using Euclidan distance in CIE L*u*v* color space and Ward's linkage) and displays the resulting dendrogram. The `plotpc` function uses the same distance metric to compute and plot principal components.

Value

In general, these functions are used for their side-effect (producing plots) rather than for their return values. In most cases, they invisibly return the color set with which they were invoked. The barplot-based functions (`swatch`, `ranswatch`, `swatchHue`, and `swatchLuminance`), however, return the vector of bar-centers, which can be used to add other information to the plot. The `plothc` function returns the dendrogram, and the `plotpc` function returns the principal components object.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

See Also

`palette36.colors`

Examples

```
data(alphabet)
rancurves(alphabet)
ranpoints(alphabet)
uvscatter(alphabet)
luminance(alphabet)
plothc(alphabet)
p3d(alphabet, cex.symbols = 2)
swatch(alphabet)
swatchHue(alphabet)
swatchLuminance(alphabet)
ranswatch(alphabet)
```

palette36 *A 36-Color Palette*

Description

A palette composed of 36 distinctive colors.

Usage

```
data(palette36)
```

Format

A character string of length 36.

Details

A character vector containing hexadecimal color representations of 36 distinctive colors that are well separated in the CIE L*u*v* color space. Each color is assigned a name from the ISCC-NBS standard.

Source

The color palette was generated using the `createPalette` function with three seed colors: ebony ("#474747"), iron ("#E2E2E2"), and red ("#F70000").

See Also

`createPalette`, `isccNames`.

Examples

```
data(palette36)
palette36
```

palettes *Polychrome Color Palettes*

Description

Five color palettes each containing at least 22 different, distinguishable colors.

Usage

```
kelly.colors(n = 22)
glasbey.colors(n = 32)
green.armytage.colors(n = 26)
palette36.colors(n = 36)
alphabet.colors(n = 26)
light.colors(n = 24)
dark.colors(n = 24)
```

Arguments

n	An integer; the number of colors desired.

Details

Kenneth Kelly, a physicist who worked at the United States National Bureau of Standards and chaired the Inter-Society Color Council Subcommittee on Color Names, made one of the earliest attempts to find a set of colors that could be easily distinguished when used in graphs. The `kelly.colors` function produces a palette from the 22 colors that he produced, using his color names. These are ordered so that the optimal contrast for any palette with fewer than 22 colors can be selected from the top of his list.

Glasbey and colleagues used a sequential search algorithm in CIE LAB color space to create a palette of 32 well-separated colors.

Paul Green-Armytage described a study growing out of a workshop held by the Colour Society of Australia in 2007 to test whether an alphabet composed of 26 distinguishable colors would serve in place of the usual symbols of the English alphabet. Each color is given a name starting with a different letter of the alphabet, which was found to make it easier for people to learn the association and read sentences written in color. The `green.armytage.colors` function produces palettes from his final color set, arranged in "alphabetical" order rather than by maximum contrast.

Carter and Carter followed Kelly's article with a study that showed that "perceptual distinguishability" of colors was related to their Euclidean distance in the L*u*v* color space coordinates, as defined by the International Commisision on Illumination (CIE). They also found that distinguishability falls off rapidly when the distance is less than about 40 L*u*v* units. We implemented a palette-construction algorithm based on this idea. The `palette36.colors` function returns palettes from the resulting list of 36 colors, with names assigned using the ISCC-NSB standard.

The `alphabet.colors` function uses the first 26 colors from "`palette36`" but assigns them names beginning with different letters of the English alphabet and reorders them accordingly.

The `light.colors` and `dark.colors` functions use one of the two 24-color palettes (`Light24` or `Dark24`) customized to limit the luminance range.

Value

Each function returns a character vector of hexadecimal color values (such as "#EA9399"). Each color is assigned a name (such as "Strong_Pink"). The default value is the maximum number of colors available from the individual palette.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

References

Kelly KL. Twenty-Two Colors of Maximum Contrast. Color Eng., 1965; 3:26–7.

Green-Armytage, P. A Colour Alphabet and the Limits of Colour Coding. Colour: Design and Creativity, 2010; 10:1–23.

Carter RC, Carter EC. High-contrast sets of colors. Applied Optics, 1982; 21(16):2936–9.

See Also

createPalette

Examples

```
palette36.colors(5)
kelly.colors(5)
alphabet.colors(7)
glasbey.colors(9)
green.armytage.colors(3)
light.colors(6)
dark.colors(11)
```

---

sortByHue *Sorting Palettes*

Description

Functions to sort palettes; potentially useful for combining existing palettes to create new ones.

Usage

```
sortByHue(colorset)
sortByLuminance(colorset)
```

Arguments

colorset	a character vector containing hexadecimal color values.

Details

These functions take a palette as input, sort it either by the hue or by the luminance, and return the result. One possibnle aplication would be to combine "dark" and "light" palettes to generate larger version of the RColorBrewer "Paired" palette.

Value

Returns a new color set (i.e., a palette, implemented as a character string containing the hex values of color), after sorting.

Author(s)

Kevin R. Coombes <krc@silicovore.com>

Examples

```
D <- dark.colors(24)
L <- light.colors(24)
X <- sortByHue(c(D,L))
names(X) <- colorNames(X)
X <- X[!duplicated(names(X))]
swatch(X)
Y <- sortByLuminance(X)
swatch(Y)
```

Index