Package 'plotwidgets'

September 6, 2016

Type Package

Title Spider Plots, ROC Curves, Pie Charts and More for Use in Other Plots

Version 0.4

Date 2016-09-06

Author January Weiner

Maintainer January Weiner <january.weiner@gmail.com>

Description Small self-contained plots for use in larger plots or to delegate plotting in other functions. Also contains a number of alternative color palettes and HSL color space based tools to modify colors or palettes.

License GPL (>= 2.0)

LazyData false

Imports methods,graphics

Suggests testthat,knitr

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-09-06 14:53:55

R topics documented:

`colorConversions` *Convert colors from and to RGB and HSL formats*

Description

Convert colors from and to RGB and HSL formats

Usage

```
col2rgb.2(col)

col2hsl(col)

hsl2col(hsl)

rgb2col(rgb)

rgb2hsl(rgb)

hsl2rgb(hsl)
```

Arguments

`col`	a character vector with colors to convert (palette)
`hsl`	a numeric matrix with three or four rows (hue, saturation, luminosity and alpha)
`rgb`	a numeric matrix with three or four rows (red, green, blue and alpha)

Details

These functions convert between RGB and HSL color spaces, and character vectors which contain color names or hash-encoded RGB values ("#FFCC00").

All functions support an alpha channel. For example, unlike the grDevices::col2rgb, col2rgb.2 returns a matrix with four rows: three for R, G and B channels and one for the alpha channel.

Value

col2rgb.2 and col2hsl return a four-row matrix. rgb2col and hsl2col return a character vector.

Functions

- `col2rgb.2`: Convert a character vector of color names (palette) to a matrix with RGB values
- `col2hsl`: Convert a character vector of color names (palette) to a matrix with HSL values
- `hsl2col`: Convert hsl matrix (3 or 4 row) to character vector of color names
- `rgb2col`: Convert rgb matrix (3 or 4 row) to character vector of color names
- `rgb2hsl`: Convert a 3- or 4-row matrix of RGB(A) values to a matrix of HSL(A) values
- `hsl2rgb`: Convert a matrix of HSL values into a matrix of RGB values

See Also

modCol, modhueCol, darkenCol, saturateCol

Examples

```
haze <- plotPals("haze")
col2rgb(haze)
col2hsl(haze)
```

---

listPlotWidgets *List available plot widgets*

Description

List available plot widgets

Usage

```
listPlotWidgets()
```

Details

Simply print out all available wigdets. A list containing the widget function is returned invisibly.

Value

Invisibly returns a named list of widget functions

Examples

```
listPlotWidgets()
```

---

modCol *Modify colors*

Description

Modify colors by shading, saturating and changing hue

Usage

```
modCol(col, darken = 0, saturate = 0, modhue = 0)

saturateCol(col, by = 0)

darkenCol(col, by = 0)

modhueCol(col, by = 0)

contrastcol(col, alpha = NULL)
```

Arguments

col	a character vector of colors (palette) to modify – a character vector
darken	Use negative values to lighten, and positive to darken.
saturate	Use negative values to desaturate, and positive to saturate
modhue	Change the hue by a number of degrees (0-360)
by	parameter for the saturateCol, darkenCol and modhueCol functions
alpha	alpha value (from 0, transparent, to 255, fully opaque)

Details

This function use the HSL (hue, saturation, luminosity) scheme to modify colors in a palette.

modCol is just a wrapper for the other three functions allowing to modify three parameters in one go.

saturateCol, darkenCol and modhueCol modify the saturation, luminosity and hue in the HSL color model.

contrastcol() returns black for each light color (with L > 0.5) and white for each dark color (with L < 0.5).

Value

a character vector containing the modified palette

Functions

- saturateCol: Change the saturation of a color or palette by a fraction of "by"
- darkenCol: Modify the darkness of a color or palette (positve by - darken, negative by – lighten)
- modhueCol: Modify the hue of a character vector of colors by by degrees
- contrastcol: Return white for dark colors, return black for light colors

Examples

```
plot.new()
## Loop over a few saturation / lightess values
par(usr=c(-0.5, 0.5, -0.5, 0.5))
v <- c(10, 9, 19, 9, 15, 5)
pal <- plotPals("zeileis")
for(sat in seq.int(-0.4, 0.4, length.out=5)) {
  for(lgh in seq.int(-0.4, 0.4, length.out=5)) {
    cols <- saturateCol(darkenCol(pal, by=sat), by=lgh)
    wgPlanets(x=sat, y=lgh, w=0.16, h=0.16, v=v, col=cols)
  }
}
axis(1)
axis(2)
title(xlab="Darkness (L) by=", ylab="Saturation (S) by=")

## Now loop over hues
a2xy <- function(a, r=1, full=FALSE) {
  t <- pi/2 - 2 * pi * a / 360
  list( x=r * cos(t), y=r * sin(t) )
}

plot.new()
par(usr=c(-1,1,-1,1))
hues <- seq(0, 360, by=30)
pos <- a2xy(hues, r=0.75)
for(i in 1:length(hues)) {
  cols <- modhueCol(pal, by=hues[i])
  wgPlanets(x=pos$x[i], y=pos$y[i], w=0.5, h=0.5, v=v, col=cols)
}

pos <- a2xy(hues[-1], r=0.4)
text(pos$x, pos$y, hues[-1])
```

`plotPals` *Return all or selected color palettes*

Description

Return all or selected color palettes from the plotwidget palette set

Usage

```
plotPals(pal = NULL, alpha = 1)
```

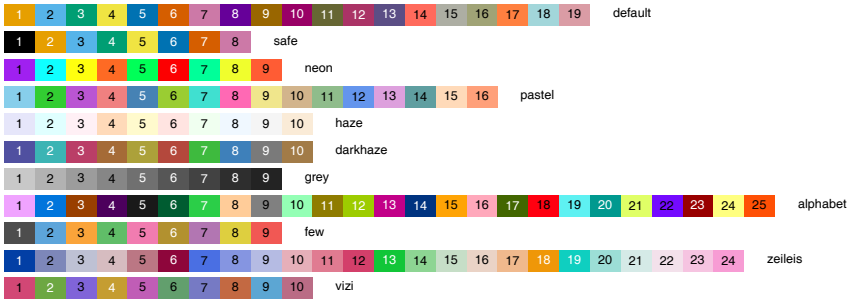
Arguments

`pal`	Name of the palette(s) to return
`alpha`	Control transparency - set alpha channel to alpha (0 - fully transparent, 1 - fully opaque)

Details

The plotwidgets package contains a number of predefined palettes, different from those in RColorBrewer.

- `default`: a standard, relatively safe (see below) palette
- `safe`: safe for color blind persons, based on Wang B. "Points of view: Color blindeness", Nature Methods 8, 441(2011)
- `neon`: a bright palette suitable for drawing on dark backgrounds
- `pastel`: a dimmed pastel palette
- `haze`: very delicate pastel colors
- `dark`: same hues as haze, but much darker
- `grey`: different shades of grey
- `alphabet`: based on "A colour alphabet..." by Paul Green-Armytage
- `few`: a palette based on Stephen Few's book
- `zeileis`: based on Zeileis et al. 2009
- `vizi`: automatically generated palette



You can get all the names of palettes with names(plotPals()), and showcase them with showPalettes(). Furthermore, you can use the `pal` parameter to `plotwidgetGallery` to see how this palette looks like with different plot widgets.

Value

Either a list of palettes, or (if only one palette was selected) a character vector with colors

See Also

`col2rgb.2`, `rgb2col`, `hsl2col`, `col2hsl`, `modCol`, `modhueCol`, `darkenCol`, `saturateCol`

Examples

```
safe <- plotPals("safe") # colorblind-safe palette
plotwidgetGallery(pal=safe)
```

`plotwidgetGallery` *A showcase of all plot types available in plotwidgets*

Description

A showcase of all plot types available in plotwidgets

Usage

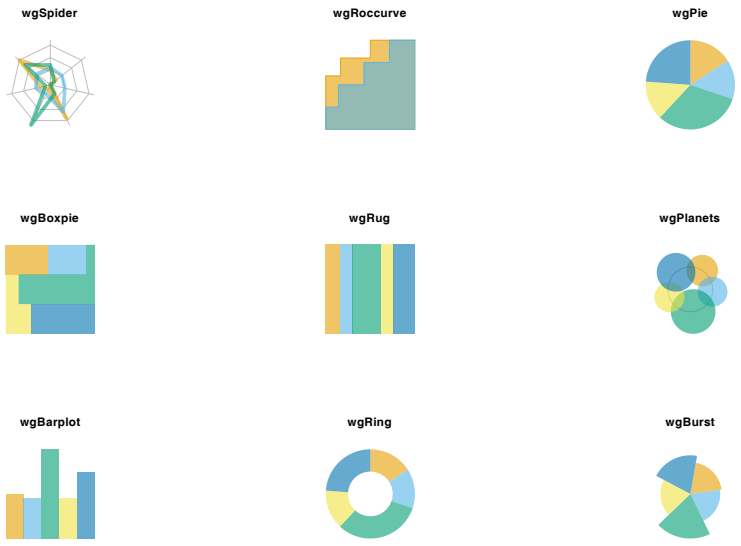
```
plotwidgetGallery(theme = "default", pal = NULL, ...)
```

Arguments

`theme`	sets both the palette and a suitable bakckground (e.g. dark for "neon")
`pal`	color palette to use (a character vector with colors)
`...`	all subsequent parameters will be passed to the plotWidget() function.

Details

plotwidgetGallery() simply draws all available plot widgets on a single plot.



Value

Invisibly returns the example data used to generate the plots

Examples

```
plotwidgetGallery()
## automatically set black bg
plotwidgetGallery(theme="neon")
## yuck, ugly:
plotwidgetGallery(pal=c("red", "#FF9900", "blue", "green", "cyan", "yellow"))
## much better:
plotwidgetGallery(pal=plotPals("pastel", alpha=0.8))
```

---

plotwidgets *Plot widgets*

Description

A collection of plotting widgets (pie charts, spider plots etc.)

Usage

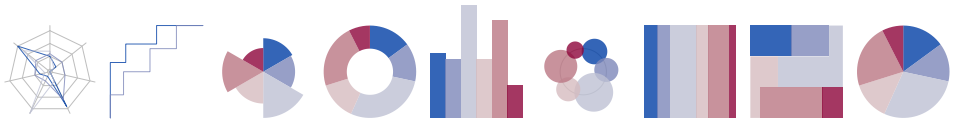
```
plotWidget(type = "pie", x = 0.5, y = 0.5, w = 1, h = 1, v,
  col = NULL, border = NA, new = FALSE, aspect = 1, ...)

wgPie(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL, border = NA,
  new = FALSE, res = 100, aspect = 1, adj = 0, labels = NULL,
  label.params = NULL)

wgRing(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL, border = NA,
  new = FALSE, res = 100, aspect = 1, adj = 0, start = 0.5,
  labels = NULL, label.params = NULL)

wgPlanets(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL,
  border = NA, new = FALSE, res = 100, aspect = 1, adj = 0,
  labels = NULL, label.params = NULL)

wgBurst(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL, border = NA,
  new = FALSE, res = 100, aspect = 1, adj = 0, max = NULL,
  labels = NULL, label.params = NULL)

wgRug(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL, border = NA,
  labels = NULL, label.params = NULL, new = FALSE, aspect = NULL,
  horizontal = TRUE, rev = FALSE)

wgBarplot(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL,
  border = NA, labels = NULL, label.params = NULL, new = FALSE,
  aspect = NULL, max = NULL)

wgBoxpie(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL, border = NA,
  labels = NULL, label.params = NULL, new = FALSE, aspect = 1,
```

```
  grid = 3)

wgRoccurve(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL,
  border = NA, new = FALSE, fill = FALSE, lwd = 1, aspect = 1)

wgSpider(x = 0.5, y = 0.5, w = 1, h = 1, v, col = NULL, border = NA,
  new = FALSE, min = NA, max = NA, tick.labels = FALSE, fill = FALSE,
  lwd = 1, aspect = 1)
```

Arguments

type	what kind of plot. To list all available plot types, use `listPlotWidgets`. Instead of the canonical name such as "wgSpider", you can simply use "spider"
x, y	coordinates at which to draw the plot
w, h	width and height of the plot
v	sizes of the slices
col	character vector with colors (palette). A default palette is chosen if the argument is NULL
border	color of the border. Use NA (default) for no border, and NULL for a border in the par("fg") color. No effect in spider plots.
new	Whether to call plot.new() before drawing the widget
aspect	the actual ratio between screen width and screen height. Parameters w and h will be scaled accordingly.
...	Any further arguments passed to plotting functions
res	for pies, rings, planets and bursts: resolution (number of polygon edges in a full circle)
adj	for spider plots, burst plots, planet plots and pies: adjust the start by "adj" degrees. For example, to turn the plot clockwise by 90 degrees, use adj=90
labels	character vector
label.params	a named list with arguments passed to text() for drawing labels (e.g. cex, col etc.)
start	For ring plots, the inner radius of the ring
max	For burst plots – a maximum value used to scale other values to make different plots comparable. For spider plots: the maximum of the scale on spider plot spikes.
horizontal	for rug plots: whether the stacked bar should be horizontal (default), or vertical
rev	logical, for rug plots: right to left or top to bottom rather than the default left to right or bottom to top
grid	boxpie only: the grid over which the areas are distributed. Should be roughly equal to the number of areas shown.
fill	for spider plots and roc curves: whether to draw filled polygons rather than lines
lwd	Line width to use
min	For spider plots: the minimum of the scale on spider plot spikes
tick.labels	For spider plots: labels for the scale ticks

Details



A widget here is a mini-plot which can be placed anywhere on any other plot, at an area specified by the center point coordinates x and y, and width and height. The widgets do not influence the layout of the plot, or change any parameters with `par()`; it only uses basic graphic primitives for maximal compatibility.

The `plotWidget()` function is a generic to call the available widgets from one place. Any widget-specific parameters are passed on through the ellipsis (...). See the documentation for specific widgets for more information. Use the functions `listPlotWidgets` and `plotwidgetGallery` to see available widgets.

The pie function draws a simple pie chart at specified coordinates with specified width, height and color. The rug function draws a corresponding rug plot, while boxpie creates a "rectangular pie chart" that is considered to be better legible than the regular pie.

Functions

- `wgPie`: `pie()` draws a pie chart with width w and height h at coordinates (x,y). The angle width of the slices is taken from the numeric vector v, and their color from the character vector col. Note that one of the main goals of the plotwidget package is to give sufficient alternatives to pie charts, hoping to help eradicate pie charts from the surface of this planet.
- `wgRing`: A pie with the center removed.
- `wgPlanets`: Produces a circular arrangement of circles, which vary in their size proportionally to the values in `v`.
- `wgBurst`: Burst plots are similar to pies. However, instead of approximating numbers by arc length, wgBurst approximates by area, making the pie slices stand out from the plot in the process.
- `wgRug`: A stacked, horizontal or vertical bar plot.
- `wgBarplot`: A minimalistic bar plot. Use the `max` paramter to scale the bars on several different widgets to the same value.
- `wgBoxpie`: Rectangular pies are thought to represent information better than pies. Here, the values in v correspond to areas rather than angles, which makes it easier to interpret it visually.
- `wgRoccurve`: This is one of two wigets that take use a different data type. `wgRoccurve` takes either a vector of T/F values, or a list of such vectors, and draws a ROC curve for each of these vectors.
- `wgSpider`: Spider plots can illustrate multivariate data. Different spikes may illustrate different variables, while different lines correspond to different samples – or vice versa. Consequently, wgSpider accepts either a vector (for a single line) or a matrix (in which each column will correspond to a single line on the plot). The length of the vector (or the number of rows in the matrix) corresponds to the number of spikes on the spider plot.

See Also

wgPie, wgBoxpie, wgSpider, wgRug, wgBurst, wgRing, wgBarplot, wgPlanets, wgRoccurve

Examples

```
# demonstration of the three widgets
plot.new()
par(usr=c(0,3,0,3))
v <- c(7, 5, 11)
col <- plotPals("safe")
b <- "black"
wgRug(0.5, 1.5, 0.8, 0.8, v=v, col=col, border=b)
wgPie(1.5, 1.5, 0.8, 0.8, v=v, col=col, border=b)
wgBoxpie(2.5, 1.5, 0.8, 0.8, v=v, col=col, border=b)

# using pie as plotting symbol
plot(NULL, xlim=1:2, ylim=1:2, xlab="", ylab="")
col <- c("#cc000099", "#0000cc99")
for(i in 1:125) {
  x <- runif(1) + 1
  y <- runif(1) + 1
  wgPie( x, y, 0.05, 0.05, c(x,y), col)
}

# square filled with box pies
n <- 10
w <- h <- 1/(n+1)
plot.new()
for(i in 1:n) for(j in 1:n)
 wgBoxpie(x=1/n*(i-1/2), y=1/n*(j-1/2), w, h,
 v=runif(3), col=plotPals("zeileis"))
```

---

showPals *Demonstrate selected palettes*

Description

Show a plot demonstrating all colors in the provided palettes

Usage

```
showPals(pal = NULL, numbers = T)
```

Arguments

pal	Either a character vector of colors or a list of character vectors
numbers	On each of the colors, show a number

Examples

```
## Show all palettes in plotwidget
showPals(plotPals())

## Show just a few colors
showPals(c("red", "green", "blue"))
```

Index