Package 'PlotTools'

May 7, 2026

Title Extended Tools for Continuous Legends, Polygon Manipulation, and Visual Display of Categorical Data

Version 0.4.0

URL https://ms609.github.io/PlotTools/, https://github.com/ms609/PlotTools/

BugReports https://github.com/ms609/PlotTools/issues/

License GPL (>= 2)

Depends R (>= 3.5)

Description Annotate plots with legends for continuous variables and colour spectra using the base graphics plotting tools; and manipulate irregular polygons. Includes palettes for colour-blind viewers.

Suggests knitr, rmarkdown, sp, spelling, testthat (>= 3.0.0), vdiffr (>= 1.0.0),

Config/Needs/check rcmdcheck, testthat (>=3.0.4), vdiffr (>= 1.0.0)

Config/Needs/coverage covr

Config/Needs/metadata codemeta

Config/Needs/revdeps revdepcheck

Config/Needs/website pkgdown

Config/testthat/edition 3

Config/testthat/parallel false

Language en-GB

Encoding UTF-8

RoxygenNote 7.3.3

LazyData true

NeedsCompilation no

Author Martin R. Smith [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-5660-1727>), Martin Krzywinski [ctb] (ORCID: <https://orcid.org/0009-0005-5785-3631>)

Maintainer Martin R. Smith <martin.smith@durham.ac.uk>

Repository CRAN

Date/Publication 2026-01-29 15:40:09 UTC

Contents

`cbPalettes` *Palettes compatible with colour blindness*

Description

Colour palettes whose elements are chosen to be distinguishable by colour blind audiences.

Usage

```
cbPalette8

cbPalette12

cbPalette15

cbPalette24

altPalette8

altPalette12

altPalette15

altPalette24
```

Format

Named character vectors listing RGB values for each colour.

An object of class `character` of length 8.

An object of class `character` of length 12.

An object of class `character` of length 15.

An object of class `character` of length 24.

An object of class `character` of length 8.

An object of class `character` of length 12.

An object of class `character` of length 15.

An object of class `character` of length 24.

Details

Palettes are named according to the number of distinct colours they contain.

Each palette is arranged in three groups, intended to achieve perceptual luminance uniformity for deuteranopic viewers (see source for details), from dark to light.

For each `cbPalette`, an `altPalette` provides a hue selected from the many that are perceptually indistinguishable for deuteranopic viewers.

Source

https://mk.bcgsc.ca/colorblind/palettes.mhtml

See Also

`palette.colors(8)` provides an eight-colour Okabe-Ito colour-blind palette.

Examples

```
par(mfrow = c(4, 1), mar = rep(1, 4))

data("cbPalette8")
plot.new()
plot.window(xlim = c(1, 8), ylim = c(-1, 1))
text(1:8, 1, col = cbPalette8)
points(1:8, rep(0, 8), bg = cbPalette8, col = altPalette8,
       pch = 22, cex = 2)

data("cbPalette12")
plot.new()
plot.window(xlim = c(1, 12), ylim = c(-1, 1))
text(1:12, 1, col = cbPalette12)
points(1:12, rep(0, 12), bg = cbPalette12, col = altPalette12,
       pch = 22, cex = 2)

data("cbPalette15")
plot.new()
plot.window(xlim = c(1, 15), ylim = c(-1, 1))
text(1:15, 1, col = cbPalette15)
points(1:15, rep(0, 15), bg = cbPalette15, col = altPalette15,
       pch = 22, cex = 2)

plot.new()
plot.window(xlim = c(1, 25), ylim = c(-1, 1))
text(1:24, 1, col = cbPalette24)
points(1:24, rep(0, 24), bg = cbPalette24, col = altPalette24,
```

```
pch = 22, cex = 2)
```

`Col2Hex` *Colour to hexadecimal conversion*

Description

Convert R colour to hexadecimal representation.

Usage

```
Col2Hex(col, alpha = FALSE)
```

Arguments

`col`	vector of any of the three kinds of R color specifications, i.e., either a color name (as listed by `colors()`), a hexadecimal string (see Details), or a positive integer `i` meaning `palette()[i]`.
`alpha`	logical value indicating whether the alpha channel (opacity) values should be returned.

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
Col2Hex(1:3)
Col2Hex(c("peachpuff", "blue"), TRUE)
```

`Polygon-Geometry` *Polygon geometry*

Description

Geometry functions for irregular polygons.

Usage

```
PolygonArea(x, y = NULL, positive = TRUE)

PolygonCentre(x, y = NULL)

PolygonCenter(x, y = NULL)

GrowPolygon(x, y = NULL, buffer = 0)
```

Arguments

x, y	Vectors containing the coordinates of the vertices of the polygon.
positive	If vertices are specified in an anticlockwise direction, the polygon will be treated as a hole, with a negative area, unless positive is set to TRUE. Vertices specified in a clockwise sequence always yield a positive area.
buffer	Numeric specifying distance by which to grow polygon.

Value

PolygonArea() returns the area of the specified polygon.

PolygonCentre() returns a single-row matrix containing the *x* and *y* coordinates of the geometric centre of the polygon.

GrowPolygon() returns coordinates of the vertices of polygon after moving each vertex buffer away from the polygon's centre.

Functions

- PolygonArea(): Calculate the area of an irregular polygon
- PolygonCentre(): Locate the centre of a polygon
- GrowPolygon(): Enlarge a polygon in all directions

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
x <- c(-3, -1, 6, 3, -4)
y <- c(-2, 4, 1, 10, 9)
plot(x, y, frame.plot = FALSE)
polygon(x, y)
PolygonArea(x, y)
points(PolygonCentre(x, y), pch = 3, cex = 2)
polygon(GrowPolygon(x, y, 1), border = "darkgreen",
        xpd = NA # Allow drawing beyond plot border
       )

# Negative values shrink the polygon
polygon(GrowPolygon(x, y, -1), border = "red")
```

SpectrumLegend *Produce a legend for continuous gradient scales*

Description

Prints an annotated vertical bar coloured according to a continuous palette.

Usage

```
SpectrumLegend(
  x = "topright",
  ...,
  palette,
  legend,
  lty = 1,
  lwd = 4,
  bty = "o",
  adj = if (horiz) c(0.5, 0.5) else c(0, 0.5),
  horiz = FALSE,
  lend = "butt",
  cex = 1,
  seg.len = 1
)

SizeLegend(
  x = "topright",
  ...,
  legend = character(0),
  width = c(0, 1),
  palette = par("col"),
  scale = c("pch", "lwd"),
  lty = 0,
  lwd = 4,
  bty = "o",
  adj = if (horiz) c(0.5, 0.5) else c(0, 0.5),
  horiz = FALSE,
  lend = "butt",
  cex = 1,
  seg.len
)
```

Arguments

x, horiz, adj, seg.len, ...	Additional parameters to legend().
palette	Colour palette to depict. Specify either a vector of colours, or a function such that palette(n) returns a vector of *n* colours.

legend	Character vector with which to label points on palette. Note that, in a vertical legend, values will be printed from top down; use rev() to reverse the order.
lwd, lty, lend	Additional parameters to segments(), controlling line style. Use lend = "butt" (the default) if palette is semitransparent, to avoid artefacts.
bty	Character specifying the type of box to be drawn around the legend. The allowed values are "o" (the default) and "n".
cex	Character expansion factor relative to current par("cex").
width	Vector of length two specifying width of legend bar at base and top.
scale	Character string specifying whether width = 1 corresponds to: "pch", the size of a plotting symbol with pch = 1; "lwd", the width of a line with lwd = 1.
col	Colour used for the width bar.

Details

This convenience function is not yet very customizable; do file a GitHub issue if you would value additional functionality.

Note that the bg parameter to specify the background colour for the legend box is not presently supported in vertical legends. For use in vertical legends, open a GitHub issue.

Value

A list, returned invisibly, with components:

- rect A list with components:
  - w, h: positive numbers giving *w*idth and *h*eight of the legend's box.
  - left, top: x and y coordinates of the upper left corner of the box.
- text: A list with components x, y, numeric vectors of length length(legend), giving the x and y coordinates of the legend's text(s).

Author(s)

Martin R. Smith (martin.smith@durham.ac.uk)

Examples

```
plot(0:1, 0:1, type = "n", frame.plot = FALSE,
     xlab = "x", ylab = "y")

SpectrumLegend("bottomright", legend = c("Bright", "Middle", "Dark"),
               palette = heat.colors(32L), lwd = 5,
               inset = 0.05, # Inset from plot margin
               title = "Brightness")
SpectrumLegend("topright", horiz = TRUE,
               legend = seq(1, 9, by = 2), palette = 1:8)
SizeLegend(
  "topleft", inset = 0.05, width = c(0, 2),
  title = "Width",
  legend = c("max", ".", ".", "min"),
```

```
  palette = topo.colors, # Associate with a colour scale
  y.intersp = 1.5 # Vertical space between labels (also moves title)
)
SizeLegend(
  "bottomleft", horiz = TRUE, width = c(4, 1),
  legend = c("Thick", "Thin"), palette = "darkred",
  inset = 0.06 # Make space for the bar.
               # A future release may calculate this automatically
)
```

Index