Package ‘PlotTools’

October 30, 2023

Title Add Continuous Legends to Plots

Version 0.3.0

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

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

License GPL (>= 2)

Depends R (>= 3.2.0)

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

Suggests knitr, rmarkdown, sp, spelling, testthat (>= 3.0), vdiffr (>= 1.0.0),

Config/Needs/check rcmdcheck

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 7.2.3

NeedsCompilation no

Author Martin R. Smith [aut, cre, cph] (<https://orcid.org/0000-0001-5660-1727>)

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

Repository CRAN

Date/Publication 2023-10-30 08:00:02 UTC

R topics documented:

`Col2Hex` *Colour to hexadecimal conversion Convert R colour to hexadecimal representation.*

Description

Colour to hexadecimal conversion 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 of the form `"#rrggbb"` or `"#rrggbbaa"` (see `rgb`), 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