Package 'netplot'

June 30, 2024

Title Beautiful Graph Drawing

Version 0.3-0

Description A graph visualization engine that emphasizes on aesthetics at the same time providing default parameters that yield out-of-the-box-nice visualizations. The package is built on top of 'The Grid Graphics Package' and seamlessly work with 'igraph' and 'network' objects.

Depends R (>= 3.4.0), grid

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Imports graphics, grDevices, igraph, network, sna, stats

Suggests knitr, rmarkdown, markdown, igraphdata, intergraph, ggraph, gridGraphics, ggplot2, gridExtra, gridBase, magrittr, tinytest

VignetteBuilder knitr

URL https://github.com/USCCANA/netplot

BugReports https://github.com/USCCANA/netplot/issues

NeedsCompilation no

Author George Vega Yon [aut, cre] (<https://orcid.org/0000-0002-3171-0844>), Porter Bischoff [aut] (<https://orcid.org/0009-0004-6742-6281>)

Maintainer George Vega Yon <g.vegayon@gmail.com>

Repository CRAN

Date/Publication 2024-06-30 20:00:02 UTC

Contents

`colorkey` *Function to create a color key*

Description

Function to create a color key

Usage

```
colorkey(
  x0,
  y0,
  x1,
  y1,
  cols = c("white", "steelblue"),
  tick.range = c(0, 1),
  tick.marks = seq(tick.range[1], tick.range[2], length.out = 5L),
  label.from = NULL,
  label.to = NULL,
  nlevels = 100,
  main = NULL,
  relative = TRUE,
  tick.args = list(),
  label.args = list(),
  main.args = list()
)
```

Arguments

`x0`, `x1`, `y0`, `y1`	Numeric scalars. Coordinates of the lower left and upper right points where the color key will be drawn as proportion of the plotting region.
`cols`	Character scalar. Colors specifications to create the color palette.
`tick.range`, `tick.marks`	Numeric vectors specifying the range and the tickmarks respectively.
`label.from`, `label.to`	Character scalar. Labels of the lower and upper values of the color key.

`nlevels`	Integer scalar. Number of levels to extrapolate.
`main`	Character scalar. Title of the colorkey.
`relative`	Logical scalar. When `TRUE` the color key is drawn relative to the plotting region area taking `x0, x1, y0, y1` as relative location.
`tick.args, label.args, main.args`	Lists of arguments passed to graphics::text for drawing ticks, labels and main respectively.

Value

NULL.

Examples

```
set.seed(22231)

# A random figure
dat <- matrix(runif(100*3), ncol = 3)
col <- colorRamp2(c("blue", "white", "red"))

plot(
  dat[,1], dat[,2],
  col = rgb(col(dat[,3]), maxColorValue=255),
  cex=2, pch=20
  )

# Pretty color key
colorkey(
  x0 = .60, y0 = .80,
  x1 = .95, y1 = .95,
  cols = c("blue", "white", "red"),
  main = "Some color scale"
)
```

`colorRamp2` *A faster implementation of grDevices::colorRamp for linear interpolation.*

Description

A faster implementation of grDevices::colorRamp for linear interpolation.

Usage

```
colorRamp2(x, alpha = TRUE, thresholds = NULL)
```

Arguments

x	A vector of colors.
alpha	Logical scalar. When TRUE This implementation of colorRamp can be 2 or more times faster than the grDevices version. It is intended for consecutive calls (i.e. in a loop) to improve performance. It is equivalent to the linear interpolation of the function colorRamp.
thresholds	A numeric vector of length length(x). Optional threshold levels so that the mixing can be different that even.

Value

A function as in grDevices::colorRamp.

Examples

```
# Creating a function for 2 colors
myf <- colorRamp2(c("black", "steelblue"))
f   <- colorRamp(c("black", "steelblue"))

plot.new()
plot.window(xlim = c(0,2), ylim = c(1, 11))

# These should be the same colors
rect(
  xleft   = 0,
  xright  = 1,
  ybottom = 1:10,
  ytop    = 2:11,
  col = rgb(myf((1:10)/10), maxColorValue = 255)
  )
rect(
  xleft   = 1,
  xright  = 2,
  ybottom = 1:10,
  ytop    = 2:11,
  col = rgb(f((1:10)/10), maxColorValue = 255)
)

# Another example setting different thresholds
myf  <- colorRamp2(c("black", "steelblue"))
myf2 <- colorRamp2(c("black", "steelblue"), thresholds=c(0, .7))

plot.new()
plot.window(xlim = c(0,2), ylim = c(1, 11))

# These should be the same colors
rect(
  xleft   = 0,
  xright  = 1,
  ybottom = 1:10,
  ytop    = 2:11,
```

```
  col = rgb(myf((1:10)/10), maxColorValue = 255)
  )
rect(
  xleft   = 1,
  xright  = 2,
  ybottom = 1:10,
  ytop    = 2:11,
  col = rgb(myf2((1:10)/10), maxColorValue = 255)
)
```

`locate_vertex` *Find a vertex in the current plot*

Description

This function is a wrapper of `grid::grid.locator()`, and provides a way to find the coordinates of a vertex in the current plot. It is useful to identify the vertex that is being clicked in a plot.

Usage

```
locate_vertex(x = NULL)
```

Arguments

`x`	An object of class `netplot`

Details

This function only works in interactive mode. Once it is called, the user can click on a vertex in the plot. The function will return the name of the vertex, the x and y coordinates and the viewport where it is located. If `x` is not specified, the last plotted `netplot` object will be used.

Value

A list with the name of the vertex, the x and y coordinates and the viewport where it is located.

Examples

```
library(igraph)
library(netplot)
set.seed(1)
x <- sample_smallworld(1, 200, 5, 0.03)

# Plotting
nplot(x)
```

```
# Clicking (only works in interactive mode)
if (interactive()) {
 res <- locate_vertex()
 print(res)
}
```

---

make_colors *Create a vector of colors for vertices and edges*

Description

Using vertex/edge attributes, these functions return vectors of colors that can be used either during the creation of the nplot object, or afterwards when changing gpar (graphical parameter) values with set_gpar.

Usage

```
make_colors(dat, categorical = FALSE, color_map = grDevices::hcl.colors)

make_edges_colors(x, eattr, ...)

make_vertex_colors(x, vattr, ...)
```

Arguments

dat	A vector of data to generate the color from.
categorical	Logical. When TRUE sets the colors as categories.
color_map	A function to generate a palette.
x	A graph of class network or igraph.
...	Further arguments passed to make_colors.
vattr, eattr	Character. Names of either vertex or edge variables to be used for generating the colors.

Details

If no attribute is provided, then by defaul the colors are set according to indegree.

x can be either a graph of class igraph or network.

Value

A vector of colors with the attribute color_map. The color map used to generate the colors.

Examples

```
data(UKfaculty, package="igraphdata")
col <- make_vertex_colors(UKfaculty, "Group")

if (require(magrittr)) {

  nplot(UKfaculty) %>%
    set_vertex_gpar("core", fill = col, col=col) %>%
    set_vertex_gpar("frame", fill = col, col=col, alpha=.7) %>%
    set_edge_gpar(col="gray50", fill="gray50", alpha=.5)
}
```

`netplot-formulae` *Formulas in* `netplot`

Description

Edge colors in both `nplot()` and `set_edge_gpar()` can be specified using a formula based on `ego()` and `alter()` (source and target). This way the user can set various types of combination varying the mixing of the colors, the alpha levels, and the actual mixing colors to create edge colors.

Usage

```
color_formula(x, col, alpha, env, type, mix = 1, postfix = NULL)

ego(...)

alter(...)
```

Arguments

`x`	An object of class netplot.
`col`	Any valid color. Can be a single color or a vector.
`alpha`	Number. Alpha levels
`env`, `type`, `postfix`	For internal use only.
`mix`	Number. For mixing colors between `ego` and `alter`
`...`	Passed to `color_formula`.

Value

Nothing. These functions are called internally when using formulas. `color_formula` modifies the environment `env`.

Examples

```
if (require(gridExtra) & require(magrittr)) {
  library(igraph)
  net <- make_ring(4)

  set.seed(1)
  np <- nplot(net, vertex.color = grDevices::hcl.colors(4), vertex.size.range=c(.1, .1))
  np %<>% set_edge_gpar(lwd = 4)

  grid.arrange(
    np,
    np %>% set_edge_gpar(col =~ego + alter),
    np %>% set_edge_gpar(col =~ego(alpha=0) + alter),
    np %>% set_edge_gpar(col =~ego + alter(alpha=0)),
    np %>% set_edge_gpar(col =~ego(mix=0) + alter(mix=1)),
    np %>% set_edge_gpar(col =~ego(mix=1) + alter(mix=0))
  )
}
```

`nplot` *Plot a network*

Description

This is a description.

Usage

```
nplot(
  x,
  layout,
  vertex.size = 1,
  bg.col = "transparent",
  vertex.nsides = 10,
  vertex.color = grDevices::hcl.colors(1),
  vertex.size.range = c(0.01, 0.03, 4),
  vertex.frame.color = NULL,
  vertex.rot = 0,
  vertex.frame.prop = 0.2,
  vertex.label = NULL,
  vertex.label.fontsize = NULL,
  vertex.label.color = adjustcolor("black", alpha.f = 0.8),
  vertex.label.fontfamily = "sans",
  vertex.label.fontface = "plain",
  vertex.label.show = 0.3,
  vertex.label.range = c(5, 15),
  edge.width = 1,
  edge.width.range = c(1, 2),
```

```
  edge.arrow.size = NULL,
  edge.color = ~ego(alpha = 0.1, col = "gray") + alter,
  edge.curvature = pi/3,
  edge.line.lty = "solid",
  edge.line.breaks = 5,
  sample.edges = 1,
  skip.vertex = FALSE,
  skip.edges = FALSE,
  skip.arrows = skip.edges,
  add = FALSE,
  zero.margins = TRUE,
  edgelist
)

## S3 method for class 'igraph'
nplot(
  x,
  layout = igraph::layout_nicely(x),
  vertex.size = igraph::degree(x, mode = "in"),
  bg.col = "transparent",
  vertex.nsides = 10,
  vertex.color = grDevices::hcl.colors(1),
  vertex.size.range = c(0.01, 0.03, 4),
  vertex.frame.color = NULL,
  vertex.rot = 0,
  vertex.frame.prop = 0.2,
  vertex.label = igraph::vertex_attr(x, "name"),
  vertex.label.fontsize = NULL,
  vertex.label.color = adjustcolor("black", alpha.f = 0.8),
  vertex.label.fontfamily = "sans",
  vertex.label.fontface = "plain",
  vertex.label.show = 0.3,
  vertex.label.range = c(5, 15),
  edge.width = igraph::edge_attr(x, "weight"),
  edge.width.range = c(1, 2),
  edge.arrow.size = NULL,
  edge.color = ~ego(alpha = 0.1, col = "gray") + alter,
  edge.curvature = pi/3,
  edge.line.lty = "solid",
  edge.line.breaks = 5,
  sample.edges = 1,
  skip.vertex = FALSE,
  skip.edges = FALSE,
  skip.arrows = !igraph::is_directed(x),
  add = FALSE,
  zero.margins = TRUE,
  edgelist
)
```

```
## S3 method for class 'network'
nplot(
  x,
  layout = sna::gplot.layout.kamadakawai(x, NULL),
  vertex.size = sna::degree(x, cmode = "indegree"),
  bg.col = "transparent",
  vertex.nsides = 10,
  vertex.color = grDevices::hcl.colors(1),
  vertex.size.range = c(0.01, 0.03, 4),
  vertex.frame.color = NULL,
  vertex.rot = 0,
  vertex.frame.prop = 0.2,
  vertex.label = network::get.vertex.attribute(x, "vertex.names"),
  vertex.label.fontsize = NULL,
  vertex.label.color = adjustcolor("black", alpha.f = 0.8),
  vertex.label.fontfamily = "sans",
  vertex.label.fontface = "plain",
  vertex.label.show = 0.3,
  vertex.label.range = c(5, 15),
  edge.width = 1,
  edge.width.range = c(1, 2),
  edge.arrow.size = NULL,
  edge.color = ~ego(alpha = 0.1, col = "gray") + alter,
  edge.curvature = pi/3,
  edge.line.lty = "solid",
  edge.line.breaks = 5,
  sample.edges = 1,
  skip.vertex = FALSE,
  skip.edges = FALSE,
  skip.arrows = !network::is.directed(x),
  add = FALSE,
  zero.margins = TRUE,
  edgelist
)

## S3 method for class 'matrix'
nplot(
  x,
  layout,
  vertex.size = 1,
  bg.col = "transparent",
  vertex.nsides = 10,
  vertex.color = grDevices::hcl.colors(1),
  vertex.size.range = c(0.01, 0.03, 4),
  vertex.frame.color = NULL,
  vertex.rot = 0,
  vertex.frame.prop = 0.2,
```

```
  vertex.label = NULL,
  vertex.label.fontsize = NULL,
  vertex.label.color = adjustcolor("black", alpha.f = 0.8),
  vertex.label.fontfamily = "sans",
  vertex.label.fontface = "plain",
  vertex.label.show = 0.3,
  vertex.label.range = c(5, 15),
  edge.width = 1,
  edge.width.range = c(1, 2),
  edge.arrow.size = NULL,
  edge.color = ~ego(alpha = 0.1, col = "gray") + alter,
  edge.curvature = pi/3,
  edge.line.lty = "solid",
  edge.line.breaks = 5,
  sample.edges = 1,
  skip.vertex = FALSE,
  skip.edges = FALSE,
  skip.arrows = skip.edges,
  add = FALSE,
  zero.margins = TRUE,
  edgelist
)

## Default S3 method:
nplot(
  x,
  layout,
  vertex.size = 1,
  bg.col = "transparent",
  vertex.nsides = 10,
  vertex.color = grDevices::hcl.colors(1),
  vertex.size.range = c(0.01, 0.03, 4),
  vertex.frame.color = NULL,
  vertex.rot = 0,
  vertex.frame.prop = 0.2,
  vertex.label = NULL,
  vertex.label.fontsize = NULL,
  vertex.label.color = adjustcolor("black", alpha.f = 0.8),
  vertex.label.fontfamily = "sans",
  vertex.label.fontface = "plain",
  vertex.label.show = 0.3,
  vertex.label.range = c(5, 15),
  edge.width = 1,
  edge.width.range = c(1, 2),
  edge.arrow.size = NULL,
  edge.color = ~ego(alpha = 0.1, col = "gray") + alter,
  edge.curvature = pi/3,
  edge.line.lty = "solid",
```

```
  edge.line.breaks = 5,
  sample.edges = 1,
  skip.vertex = FALSE,
  skip.edges = FALSE,
  skip.arrows = skip.edges,
  add = FALSE,
  zero.margins = TRUE,
  ...,
  edgelist
)

## S3 method for class 'netplot'
print(x, y = NULL, newpage = TRUE, legend = TRUE, ...)
```

Arguments

x	A graph. It supports networks stored as igraph, network, and matrices objects (see details).
layout	Numeric two-column matrix with the graph layout in x/y positions of the vertices.
vertex.size	Numeric vector of length vcount(x). Absolute size of the vertex from 0 to 1.
bg.col	Color of the background.
vertex.nsides	Numeric vector of length vcount(x). Number of sizes of the vertex. E.g. three is a triangle, and 100 approximates a circle.
vertex.color	Vector of length vcount(x). Vertex HEX or built in colors.
vertex.size.range	Numeric vector of length 3. Relative size for the minimum and maximum of the plot, and curvature of the scale. The third number is used as size^rel[3].
vertex.frame.color	Vector of length vcount(x). Border of vertex in HEX or built in colors.
vertex.rot	Vector of length vcount(x) in Radians. Passed to npolygon, elevation degree from which the polygon is drawn.
vertex.frame.prop	Vector of length vcount(x). What proportion of the vertex does the frame occupy (values between 0 and 1).
vertex.label	Character vector of length vcount(x). Labels.
vertex.label.fontsize	Numeric vector.
vertex.label.color	Vector of colors of length vcount(x).
vertex.label.fontfamily	Character vector of length vcount(x).
vertex.label.fontface	See grid::gpar

`vertex.label.show`
: Numeric scalar. Proportion of labels to show as the top ranking according to `vertex.size`.

`vertex.label.range`
: Numeric vector of size 2 or 3. Relative scale of `vertex.label.fontsize` in points (see grid::gpar).

`edge.width`
: Vector of length `ecount(x)` from 0 to 1. All edges will be the same size.

`edge.width.range`
: Vector of length `ecount(x)` from 0 to 1. Adjusting width according to weight.

`edge.arrow.size`
: Vector of length `ecount(x)` from 0 to 1.

`edge.color`
: A vector of length `ecount(x)`. In HEX or built in colors. Can be `NULL` in which case the color is picked as a mixture between ego and alters' `vertex.color` values.

`edge.curvature`
: Numeric vector of length `ecount(x)`. Curvature of edges in terms of radians.

`edge.line.lty`
: Vector of length `ecount(x)`. Line types in R (e.g.- 1 = Solid, 2 = Dashed, etc).

`edge.line.breaks`
: Vector of length `ecount(x)`. Number of vertices to draw (approximate) the arc (edge).

`sample.edges`
: Numeric scalar between 0 and 1. Proportion of edges to sample.

`skip.vertex`, `skip.edges`, `skip.arrows`
: Logical scalar. When `TRUE` the object is not plotted.

`add`
: Logical scalar.

`zero.margins`
: Logical scalar.

`edgelist`
: An edgelist.

`y`, `...`
: Ignored

`newpage`
: Logical scalar. When `TRUE` calls grid::grid.newpage.

`legend`
: Logical scalar. When `TRUE` it adds a legend.

Details

When `x` is of class matrix, it will be passed to `igraph::graph_from_adjacency_matrix()`.

In the case of `edge.color`, the user can specify colors using netplot-formulae.

Value

An object of class `c("netplot", "gTree", "grob", "gDesc")`. The object has an additional set of attributes:

- `.xlim, .ylim` vector of size two with the x-asis/y-axis limits.
- `.layout` A numeric matrix of size `vcount(x) * 2` with the vertices positions
- `.edgelist` A numeric matrix, The edgelist.

In the case of `nplot.default`, an object of class `netplot` and `grob` (see grid::grob) with the following slots:

- children The main grob of the object.
- name Character scalar. The name of the plot
- .xlim and .ylim Two vectors indicating the limits of the plot
- .layout A two-column matrix with the location of the vertices.
- .edgelist A two-column matrix, an edgelist.
- .N Integer. The number of vertices.
- .M Integer. The number of edges.

The children grob contains the following two objects:

- background a grob rectangle.
- graph a gTree that contains each vertex and each edge of the figure.

See Also

nplot_base

Examples

```
library(igraph)
library(netplot)
set.seed(1)
x <- sample_smallworld(1, 200, 5, 0.03)

plot(x) # ala igraph
nplot(x) # ala netplot
```

---

nplot_base nplot *using base graphics*

Description

nplot using base graphics

Usage

```
nplot_base(
  x,
  layout = igraph::layout_nicely(x),
  vertex.size = igraph::degree(x, mode = "in"),
  bg.col = "transparent",
  vertex.nsides = 10,
  vertex.color = grDevices::hcl.colors(1),
  vertex.size.range = c(0.01, 0.03, 4),
 vertex.frame.color = grDevices::adjustcolor(vertex.color, red.f = 0.75, green.f = 0.75,
    blue.f = 0.75),
```

```
  vertex.rot = 0,
  vertex.frame.prop = 0.1,
  edge.width = NULL,
  edge.width.range = c(1, 2),
  edge.arrow.size = NULL,
  edge.color = NULL,
  edge.color.mix = 0.5,
  edge.color.alpha = c(0.1, 0.5),
  edge.curvature = pi/3,
  edge.line.lty = "solid",
  edge.line.breaks = 5,
  sample.edges = 1,
  skip.vertex = FALSE,
  skip.edges = FALSE,
  skip.arrows = skip.edges,
  add = FALSE,
  zero.margins = TRUE
)
```

Arguments

`x`	A graph. It supports networks stored as `igraph`, `network`, and matrices objects (see details).
`layout`	Numeric two-column matrix with the graph layout in x/y positions of the vertices.
`vertex.size`	Numeric vector of length `vcount(x)`. Absolute size of the vertex from 0 to 1.
`bg.col`	Color of the background.
`vertex.nsides`	Numeric vector of length `vcount(x)`. Number of sizes of the vertex. E.g. three is a triangle, and 100 approximates a circle.
`vertex.color`	Vector of length `vcount(x)`. Vertex HEX or built in colors.
`vertex.size.range`	Numeric vector of length 3. Relative size for the minimum and maximum of the plot, and curvature of the scale. The third number is used as `size^rel[3]`.
`vertex.frame.color`	Vector of length `vcount(x)`. Border of vertex in HEX or built in colors.
`vertex.rot`	Vector of length `vcount(x)` in Radians. Passed to npolygon, elevation degree from which the polygon is drawn.
`vertex.frame.prop`	Vector of length `vcount(x)`. What proportion of the vertex does the frame occupy (values between 0 and 1).
`edge.width`	Vector of length `ecount(x)` from 0 to 1. All edges will be the same size.
`edge.width.range`	Vector of length `ecount(x)` from 0 to 1. Adjusting width according to weight.
`edge.arrow.size`	Vector of length `ecount(x)` from 0 to 1.

`edge.color` A vector of length `ecount(x)`. In HEX or built in colors. Can be `NULL` in which case the color is picked as a mixture between ego and alters' `vertex.color` values.

`edge.color.mix` Proportion of the mixing.

`edge.color.alpha`
Either a vector of length 1 or 2, or a matrix of size `ecount(x)*2` with values in `[0,1]`. Alpha (transparency) levels (see details)

`edge.curvature` Numeric vector of length `ecount(x)`. Curvature of edges in terms of radians.

`edge.line.lty` Vector of length `ecount(x)`. Line types in R (e.g.- 1 = Solid, 2 = Dashed, etc).

`edge.line.breaks`
Vector of length `ecount(x)`. Number of vertices to draw (approximate) the arc (edge).

`sample.edges` Numeric scalar between 0 and 1. Proportion of edges to sample.

`skip.vertex`, `skip.edges`, `skip.arrows`
Logical scalar. When `TRUE` the object is not plotted.

`add` Logical scalar.

`zero.margins` Logical scalar.

Value

`nplot_base` returns a list with the following components:

- `vertex.coords` A list of length `N` where each element describes the geomtry of each vertex.
- `vertex.color` A vector of colors
- `vertex.frame.coords` Similar to `vertex.coords`, but for the frame.
- `vertex.frame.color` Similar to `vertex.color`, but for the frame.
- `edge.color` Vector of functions used to compute the edge colors.
- `edge.coords` Similar to `vertex.coords`, the points that describe each edge.
- `edge.arrow.coords` A list of matrices describing the geometry of the tip of the edges.
- `edge.width` A numeric vector with edges' widths.
- `xlim`, `ylim` Limits of the plot area.

See Also

nplot

Examples

```
# Same example as in nplot
library(igraph)
library(netplot)
set.seed(1)
x <- sample_smallworld(1, 200, 5, 0.03)

nplot_base(x) # ala netplot (using base)
```

nplot_legend *Add legend to a netplot object*

Description

Legends in grid graphics is a bit more complicated than in base graphics. The function nplot_legend is a wrapper of grid::legendGrob() that makes the process easier. Besides labels, the main visual arguments for the figure ar passed through the gp argument (see examples).

Usage

```
nplot_legend(
  g,
  labels,
  pch,
  gp = grid::gpar(),
  ...,
  packgrob.args = list(side = "left")
)

## S3 method for class 'netplot_legend'
print(x, y = NULL, newpage = TRUE, ...)
```

Arguments

g	An object of class netplot.
labels	Character vector of labels.
pch	See graphics::points().
gp	An object of class grid::gpar()
...	Further arguments passed to grid::legendGrob().
packgrob.args	List of arguments passed to grid::packGrob().
x	An object of class netplot_legend.
y	Ignored.
newpage	Logical scalar. When TRUE it calls grid::grid.newpage().

Value

A frame grob.

Examples

```
library(igraph)
library(netplot)
set.seed(1)
x <- sample_smallworld(1, 200, 5, 0.03)
```

```
V(x)$nsides <- sample(c(10, 4), 200, replace = TRUE)

g <- nplot(
  x,
  vertex.nsides = V(x)$nsides,
  vertex.color  = ifelse(V(x)$nsides == 4, "red", "steelblue"),
  edge.line.breaks = 5
  )

nplot_legend(
  g,
  labels = c("circle", "diamond", "edge"),
  pch    = c(21, 23, NA),
  gp     = gpar(
    fill = c("steelblue", "red", NA),
    lwd  = c(NA, NA, 1),
    col  = c(NA, NA, "purple")
    )
  )
grid.text("Legend to the left (default)", y = unit(.95, "npc"), just = "bottom")

nplot_legend(
  g,
  labels = c("circle", "diamond", "edge"),
  pch    = c(21, 23, NA),
  gp     = gpar(
    fill = c("steelblue", "red", NA),
    lwd  = c(NA, NA, 1),
    col  = c(NA, NA, "purple")
    ),
  # These two extra options set the legend to the bottom
  packgrob.args = list(side = "bottom"),
  ncol = 3
  )
grid.text("Legend bottom", y = unit(.95, "npc"), just = "bottom")
```

---

npolygon *n-sided polygons Calculate the coordinates for an nsided polygon*

Description

n-sided polygons Calculate the coordinates for an nsided polygon

Usage

```
npolygon(x = 0, y = 0, n = 6L, r = 1, d = 2 * pi/(n)/2)
```

Arguments

x, y	Numeric scalar. Origin of the polygon.
n	Integer scalar. Number of sides.
r	Numeric scalar. Radious of the polygon.
d	Numeric scalar. Starting degree in radians.

Value

A two column matrix with the coordinates to draw a n sided polygon.

Examples

```
graphics.off()
oldpar <- par(no.readonly = TRUE)

par(xpd = NA, mfrow = c(3, 3), mai = rep(0, 4))
for (n in c(2, 3, 4, 5, 6, 8, 12, 20, 50)) {

  plot.new()
  plot.window(c(-1.25,1.25), c(-1.25,1.25))

  for (i in seq(1, .0005, length.out = 200)) {
    col <- adjustcolor("tomato", alpha.f = i)
    polygon(npolygon(x=(i-1)/4, y = (i-1)/4, r = i, d = i-1, n = n),
            col = NA, border=col)
  }

  mtext(sprintf("n = %i", n), side = 1, line = -3)
}

par(oldpar)
```

---

piechart *A flexible piechart.*

Description

While similar to graphics::pie(), this function is much more flexible as it allows providing different parameters for each slice of the pie. Furthermore, it allows adding the plot to the current device, making it possible to create compound piecharts.

Usage

```
piechart(
  x,
  labels = names(x),
  radius = 1,
```

```
  doughnut = 0,
  origin = c(0, 0),
  edges = 200,
  slice.off = 0,
  init.angle = 0,
  last.angle = 360,
  tick.len = 0.1,
  text.args = list(),
  segments.args = list(),
  skip.plot.slices = FALSE,
  add = FALSE,
  rescale = TRUE,
  ...
)
```

Arguments

x	Numeric vector. Values that specify the area of the slices.
labels	Character vector of length length(x). Passed to graphics::text().
radius	Numeric vector. Radious of each slice (can be a scalar).
doughnut	Numeric scalar. Radious of each inner circle (doughnut) (can be a scalar).
origin	Numeric vector of length 2. Coordinates of the origin.
edges	Numeric scalar. Smoothness of the slices curve (can be a vector).
slice.off	Numeric vector. When !=0, specifies how much to move the slice away from the origin. When scalar is recycled.
init.angle	Numeric scalar. Angle from where to start drawing in degrees.
last.angle	Numeric scalar. Angle where to finish drawing in degrees.
tick.len	Numeric scalar. Size of the tick marks as proportion of the radius.
text.args	List. Further arguments passed to graphics::text().
segments.args	List. Further arguments passed to graphics::segments() when drawing the tickmarks.
skip.plot.slices	Logical scalar. When FALSE, slices are not drawn. This can be useful if, for example, the user only wants to draw the labels.
add	Logical scalar. When TRUE it is added to the current device.
rescale	Logical scalar. When TRUE (default), the y-coordinates of the polygons (slices), text and tickmarks will be rescaled such that the aspectt ratio is preserved, i.e. looks like a circle.
...	Further arguments passed to graphics::polygon() (see details).

Details

The function is a wrapper of graphics::polygon(), so all parameters such as color, density, border, etc. are passed directly by mapply() so that are specified one per slice. The coordinates of the slices are computed internally.

Value

A list with the following elements:

`slices`	A list of length `length(x)` with the coordinates of each slice.
`textcoords`	A numeric matrix of size `length(x)*2` with coordinates where the labels can be put at.
`alpha0`	A numeric vector of size `length(x)` with the starting degree in radians of the slice.
`alpha1`	A numeric vector of size `length(x)` with the ending degree in radians of the slice.

See Also

https://commons.wikimedia.org/wiki/File:Nightingale-mortality.jpg

Examples

```
# Example 1 ---------------------------------------------------------------
# A set of 3 nested rings rings starting at 315 deg. and ending at 270 deg.

# Values to plot
vals <- c(1,2,3,10)

# Outer (includes labels)
piechart(vals, col=grDevices::blues9[5:8], border=NA, doughnut = .5,
    radius=.75, labels=vals, init.angle = 315, last.angle = 270)

# Middle
piechart(vals, col=grDevices::blues9[3:6], border=NA, doughnut = .3,
    radius=.5, add=TRUE, init.angle = 315, last.angle = 270)

# Inner
piechart(vals, col=grDevices::blues9[1:4], border="gray", doughnut = .1,
    radius=.3, add=TRUE, init.angle = 315, last.angle = 270)

# Example 2 ---------------------------------------------------------------
# Passing values to polygon and playing with the radius and slice.off

piechart(1:10, density=(1:10)^2/2, slice.off = (1:10)/30, doughnut = .5,
  radius = sqrt(10:1),
  # Here we are setting random labels...
  labels=sapply(1:10, function(x) paste(sample(letters, x, TRUE), collapse="")),
  )
```

segments_gradient *Draw segments colored by gradients*

Description

Draw segments colored by gradients

Usage

```
segments_gradient(
  x,
  y = NULL,
  col = colorRamp2(c("transparent", "black"), TRUE),
  lend = 1,
  ...
)
```

Arguments

x, y	Coordinates passed to grDevices::xy.coords.
col	Color ramp function (see grDevices::colorRamp).
lend	Passed to graphics::segments.
...	Further arguments passed to segments.

Value

See graphics::segments.

Examples

```
set.seed(1)
x <- cbind(cumsum(rnorm(1e3, sd=.1)), cumsum(rnorm(1e3, sd=.4)))
plot(x, type="n")
segments_gradient(x)
```

set_gpar *Set/retrieve graphical parameters of a* netplot *object*

Description

Set/retrieve graphical parameters of a netplot object

Usage

```
set_gpar(x, type, element, idx, ...)

set_edge_gpar(x, element, idx, ...)

set_vertex_gpar(x, element, idx, ...)

get_vertex_gpar(x, element, ..., idx)

get_edge_gpar(x, element, ..., idx)

get_gpar(x, type, element, ..., idx, simplify = TRUE)
```

Arguments

x	An object of class netplot.
type	Character. Either "edge" or "vertex".
element	Character. If "edge", then it can be either "line" or "arrow", otherwise it can be either "core" or "frame".
idx	(optional) Integer vector. Indices of the elements to be modified. When missing, all elements are modified.
...	Parameters to be modified/retrieved. This is passed to grid::editGrob via grid::gpar.
simplify	Logical. When TRUE it tries to simplify the result. Otherwise it returns a nested list.

Details

set_edge_gpar and set_vertex_gpar are shorthands for set_gpar(type = "edge", ...) and set_gpar(type = "vertex", ...) respectively.

get_edge_gpar and get_vertex_gpar are shorthands for get_gpar(type = "edge", ...) and get_gpar(type = "vertex", ...) respectively.

Value

An object of class netplot with modified parameters.

Examples

```
library(igraph)
library(netplot)

x <- make_ring(5)

g <- nplot(x)

# Updating edge color
g <- set_edge_gpar(g, col = "gray80")
```

```
# Retrieving the color of the vertices (core)
get_vertex_gpar(g, element = "core", "fill", "lwd")
```

Index