Package 'cartography'

August 29, 2016

Title Thematic Cartography

Version 1.4.0

Date 2016-08-24

Description Create and integrate maps in your R workflow. This package allows various cartographic representations such as proportional symbols, chroropleth, typology, flows or discontinuities. In addition, it also proposes some useful features like cartographic palettes, layout (scale, north arrow, title...), labels, legends or access to cartographic API to ease the graphic presentation of maps.

License GPL-3

URL https://github.com/Groupe-ElementR/cartography/

BugReports https://github.com/Groupe-ElementR/cartography/issues/

LazyData true

Depends R (>= 2.10), sp

Imports classInt, stats, graphics, utils, methods, rgeos, rosm, raster

Suggests SpatialPosition, knitr, rmarkdown

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 5.0.1

NeedsCompilation no

Author Timothée Giraud [cre, aut],
Nicolas Lambert [aut]

Maintainer Timothée Giraud <timothee.giraud@ums-riate.fr>

Repository CRAN

Date/Publication 2016-08-26 19:02:27

R topics documented:

`carto.pal` *Build Cartographic Palettes*

Description

Build sequantial, diverging and qualitative color palettes. Diververging color palettes can be dissymetric (different number of colors in each of the two gradients).

Usage

```
carto.pal(pal1, n1, pal2 = NULL, n2 = NULL, middle = FALSE,
  transparency = FALSE)
```

Arguments

`pal1`	name of the color gradiant (see Details).
`n1`	number of colors (up to 20)
`pal2`	name of the color gradiant (see Details).
`n2`	number of colors (up to 20)
`middle`	a logical value. If TRUE, a neutral color ("#F6F6F6", almost white) between two gradients is added.
`transparency`	a logical value. If TRUE, contrasts are enhenced by adding an opacity variation.

Details

Sequential palettes:

- blue.pal
- orange.pal
- red.pal
- brown.pal
- green.pal
- purple.pal
- pink.pal
- wine.pal
- grey.pal
- turquoise.pal
- sand.pal
- taupe.pal

- kaki.pal
- harmo.pal

Qualitative palettes:

- pastel.pal
- multi.pal

Value

A vector of colors is returned.

Note

Use display.carto.all to show all palettes and use display.carto.pal to show one palette.

References

Qualitative palettes were generated with "i want hue" (http://tools.medialab.sciences-po.fr/iwanthue/) by Mathieu Jacomy at the Sciences-Po Medialab.

See Also

display.carto.pal, display.carto.all, carto.pal.info

Examples

```
# Simple gradient: blue
carto.pal(pal1 = "blue.pal" ,n1 = 20)

# Double gradient: blue & red
carto.pal(pal1 = "blue.pal", n1 = 10, pal2 = "red.pal", n2 = 10)

# Adding a neutral color
carto.pal(pal1 = "blue.pal", n1 = 10, pal2 = "red.pal", n2 = 10, middle = TRUE)

# Enhancing contrasts with transparency
carto.pal(pal1="blue.pal", n1 = 10, pal2 = "red.pal", n2 = 10, middle = TRUE,
          transparency = TRUE)

# The double gradient can be asymetric
carto.pal(pal1 = "blue.pal", n1 = 5, pal2 = "red.pal", n2 = 15, middle = TRUE,
          transparency = TRUE)

# Build and display a palette
mypal <- carto.pal(pal1 = "blue.pal", n1 = 5, pal2 = "red.pal", n2 = 15,
                   middle = TRUE, transparency = TRUE)
k <- length(mypal)
image(1:k, 1, as.matrix(1:k), col =mypal, xlab = paste(k," classes",sep=""),
      ylab = "", xaxt = "n", yaxt = "n",bty = "n")
```

carto.pal.info *Display the Names of all Cartographic Palettes*

Description

Display the names of all the available color palettes.

Usage

```
carto.pal.info()
```

Value

A vector of palettes names is returned.

See Also

carto.pal, display.carto.pal, display.carto.all

Examples

```
carto.pal.info
```

cartography *Cartography Package*

Description

The cartography package allows various cartographic representations such as proportional symbols, chroropleth, typology, flows or discontinuities. In addition it also proposes some useful features like cartographic palettes, layout (scale, north arrow, title...), labels, legends or access to cartographic API to ease the graphic presentation of maps.

A vignette contains commented scripts on how to build various types of maps with `cartography`: `vignette(topic = "cartography")`

Main functions :

- Proportional symbols maps (circles, squares, bars)
  propSymbolsLayer, propSymbolsChoroLayer, propSymbolsTypoLayer, propTrianglesLayer
- Chroropleth maps (main discretization methods are available)
  choroLayer
- Typology maps
  typoLayer
- Flow maps (proportional and classified links)
  getLinkLayer, propLinkLayer, gradLinkLayer

- Discontinuities maps (variable size and color of borders)
  getBorders, discLayer
- Cartographic palettes (palettes adapted to cartographic representation)
  carto.pal
- Layout (scale, north arrow, title...)
  layoutLayer
- Labels
  labelLayer
- Legends
  legendBarsSymbols, legendChoro, legendCirclesSymbols, legendGradLines, legendPropLines, legendPropTriangles, legendSquaresSymbols, legendTypo
- Access to cartographic API (via rosm package)
  getTiles, tilesLayer
- Irregular polygons to regular grid transformation with data handling
  getGridLayer, getGridData

`cartography.colors` *Color Palettes*

Description

List of color gradients adapted to thematic cartography.

Source

UMS RIATE

See Also

display.carto.pal, display.carto.all, carto.pal

`choroLayer` *Choropleth Layer*

Description

Plot a chorpoleth layer.

Usage

```
choroLayer(spdf, df, spdfid = NULL, dfid = NULL, var, breaks = NULL,
  method = "quantile", nclass = NULL, col = NULL, border = "grey20",
  lwd = 1, colNA = "white", legend.pos = "bottomleft",
  legend.title.txt = var, legend.title.cex = 0.8, legend.values.cex = 0.6,
  legend.values.rnd = 0, legend.nodata = "no data", legend.frame = FALSE,
  add = FALSE)
```

Arguments

spdf	a SpatialPolygonsDataFrame.
df	a data frame that contains the values to plot. If df is missing spdf@data is used instead.
spdfid	name of the identifier field in spdf, default to the first column of the spdf data frame. (optional)
dfid	name of the identifier field in df, default to the first column of df. (optional)
var	name of the numeric field in df to plot.
breaks	break values in sorted order to indicate the intervals for assigning the colors. Note that if there are nlevel colors (classes) there should be (nlevel+1) break values (see Details).
method	a discretization method; one of "sd", "equal", "quantile", "fisher-jenks","q6" or "geom" (see Details).
nclass	a targeted number of classes. If null, the number of class is automatically defined (see Details).
col	a vector of colors. Note that if breaks is specified there must be one less colors specified than the number of break.
border	color of the polygons borders.
lwd	borders width.
colNA	no data color.
legend.pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.
legend.title.txt	title of the legend.
legend.title.cex	size of the legend title.
legend.values.cex	size of the values in the legend.
legend.values.rnd	number of decimal places of the values in the legend.
legend.nodata	no data label.
legend.frame	whether to add a frame to the legend (TRUE) or not (FALSE).
add	whether to add the layer to an existing plot (TRUE) or not (FALSE).

Details

The optimum number of class depends on the number of geographical objects. If nclass is not defined, an automatic method inspired by Sturges (1926) is used : nclass = 1+3.3*log10(N), where nclass is the number of class and N is the variable length.

If breaks is used then nclass and method are not.

"sd", "equal", "quantile" and "fisher-jenks" are classIntervals methods. Jenks and Fisher-Jenks algorithms are based on the same principle and give quite similar results but Fisher-Jenks is much faster.
The "q6" method uses the following quantile probabilities: 0, 0.05, 0.275, 0.5, 0.725, 0.95, 1.
The "geom" method is based on a geometric progression along the variable values.

References

Herbert A. Sturges, « *The Choice of a Class Interval* », Journal of the American Statistical Association, vol. 21, n° 153, mars 1926, p. 65-66.

See Also

getBreaks, carto.pal, legendChoro, propSymbolsChoroLayer

Examples

```
data("nuts2006")

## Example 1
nuts2.df$unemprate <- nuts2.df$unemp2008/nuts2.df$act2008*100
choroLayer(spdf = nuts2.spdf,
           df = nuts2.df,
           var = "unemprate")

## Example 2
nuts2.df$unemprate <- nuts2.df$unemp2008/nuts2.df$act2008*100
choroLayer(spdf = nuts2.spdf,
           df = nuts2.df,
           var = "unemprate",
           method = "quantile",
           nclass = 8,
           col = carto.pal(pal1 = "turquoise.pal", n1 = 8),
           border = "grey40",
           add = FALSE,
           legend.pos = "topright",
           legend.title.txt = "Unemployement rate\n(%)",
           legend.values.rnd = 1)

## Example 3
# Compute the compound annual growth rate
nuts2.df$cagr <- (((nuts2.df$pop2008 / nuts2.df$pop1999)^(1/9)) - 1) * 100
summary(nuts2.df$cagr)
# Plot the compound annual growth rate
cols <- carto.pal(pal1 = "blue.pal", n1 = 2, pal2 = "red.pal", n2 = 4)
choroLayer(spdf = nuts2.spdf,
           df = nuts2.df,
           var = "cagr", breaks = c(-2.43,-1,0,0.5,1,2,3.1),
           col = cols,
           border = "grey40",
           add = FALSE,
           legend.pos = "topright",
```

```
            legend.title.txt = "Compound annual\ngrowth rate",
            legend.values.rnd = 2)
# Layout plot
layoutLayer(title = "Demographic Trends",
            sources = "Eurostat, 2008",
            scale = NULL,
            frame = TRUE,
            col = "black",
            coltitle = "white")
```

`coasts.spdf` *Coastline of Europe*

Description

Coastline of Europe.

Format

SpatialLinesDataFrame.

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

`countries.spdf` *Countries in the European Area*

Description

Countries in the european area.

Format

SpatialPolygonsDataFrame.

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

`discLayer` *Discontinuities Layer*

Description

This function computes and plots spatial discontinuities. The discontinuities are plotted over the layer outputed by the getBorders function. The line widths reflect the ratio between values of an indicator in two neighbouring units.

Usage

```
discLayer(spdf, df, spdfid1 = NULL, spdfid2 = NULL, dfid = NULL, var,
  method = "quantile", nclass = 4, threshold = 0.75, type = "rel",
  sizemin = 1, sizemax = 10, col = "red", legend.pos = "bottomleft",
  legend.title.txt = "legend title", legend.title.cex = 0.8,
  legend.values.cex = 0.6, legend.values.rnd = 2, legend.frame = FALSE,
  add = TRUE)
```

Arguments

`spdf`	a SpatialLinesDataFrame, as outputed by the getBorders function.
`df`	a data frame that contains the values used to compute and plot discontinuities.
`spdfid1`	first identifier of the border, default to the second column of the spdf data frame. (optional)
`spdfid2`	second identifier of the border, default to the third column of the spdf data frame. (optional)
`dfid`	identifier field in df, default to the first column of df. (optional)
`var`	name of the numeric field in df used to compute and plot discontinuities.
`method`	a discretization method; one of "sd", "equal", "quantile", "fisher-jenks","q6" or "geom" (see getBreaks Details).
`nclass`	a targeted number of classes. If null, the number of class is automatically defined (see getBreaks Details).
`threshold`	share of represented borders, value between 0 (nothing) and 1 (all the discontinuities).
`type`	type of discontinuity measure, one of "rel" or "abs" (see Details).
`sizemin`	thickness of the smallest line.
`sizemax`	thickness of the biggest line.
`col`	color of the discontinuities lines.
`legend.pos`	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.
`legend.title.txt`	title of the legend.

`legend.title.cex`
: size of the legend title.

`legend.values.cex`
: size of the values in the legend.

`legend.values.rnd`
: number of decimal places of the values in the legend.

`legend.frame`
: whether to add a frame to the legend (TRUE) or not (FALSE).

`add`
: whether to add the layer to an existing plot (TRUE) or not (FALSE).

Details

The "rel" type of discontinuity is the result of pmax(value unit 1 / value unit 2, value unit 2 / value unit 1).
The "abs" type of discontinuity is the result of pmax(value unit 1 - value unit 2, value unit 2 - value unit 1).

Value

An invisible (invisible) SpatialLinesDataFrame with the discontinuity measures is returned.

See Also

getBorders, gradLinkLayer, legendGradLines

Examples

```
data(nuts2006)
# Get borders
nuts0.contig.spdf <- getBorders(nuts0.spdf)
# GDP per capita
nuts0.df$gdpcap <- nuts0.df$gdppps2008/nuts0.df$pop2008
# Plot countries
plot(nuts0.spdf, col="#CCCCCC", lwd=1, border="white")
# Plot discontinuities
discLayer(spdf = nuts0.contig.spdf, df = nuts0.df,
          var = "gdpcap", col="red", nclass=5,
          method="quantile", threshold = 0.5, sizemin = 1,
          sizemax = 10, type = "rel", legend.frame = TRUE,
          legend.title.txt = "GDP per Capita discontinuities\n(relative)",
          legend.pos = "topright", add=TRUE)
```

---

display.carto.all *Display all Cartographic Palettes*

Description

Display all the available color palettes.

Usage

```
display.carto.all(n = 10)
```

Arguments

n	number of colors in the gradients (from 1 to 20).

See Also

carto.pal, display.carto.pal, carto.pal.info

Examples

```
display.carto.all(1)
display.carto.all(5)
display.carto.all(8)
display.carto.all(12)
display.carto.all(20)
```

---

display.carto.pal *Display one Cartographic Palette*

Description

Display one color palette.

Usage

```
display.carto.pal(name)
```

Arguments

name	name of the palette available in the package (see Details).

Details

Sequential palettes:

- blue.pal
- orange.pal
- red.pal
- brown.pal
- green.pal
- purple.pal
- pink.pal
- wine.pal

- grey.pal
- turquoise.pal
- sand.pal
- taupe.pal
- kaki.pal
- harmo.pal

Qualitative palettes:

- pastel.pal
- multi.pal

See Also

carto.pal, display.carto.all, carto.pal.info

Examples

```
display.carto.pal("orange.pal")
display.carto.pal("sand.pal")
```

`dotDensityLayer` *Dot Density layer*

Description

Plot a dot density layer.

Usage

```
dotDensityLayer(spdf, df, spdfid = NULL, dfid = NULL, var, n = NULL,
  iter = 5, pch = 1, cex = 0.15, type = "random", col = "black",
  legend.pos = "topright", legend.txt = NULL, legend.cex = 0.6,
  legend.col = "black", legend.frame = TRUE, add = TRUE)
```

Arguments

`spdf`	SpatialPointsDataFrame or SpatialPolygonsDataFrame; if spdf is a SpatialPolygonsDataFrame symbols are plotted on centroids.
`df`	a data frame that contains the values to plot. If df is missing spdf@data is used instead.
`spdfid`	id field in spdf, default to the first column of the spdf data frame. (optional)
`dfid`	id field in df, default to the first column of df. (optional)
`var`	name of the numeric field in df to plot.
`n`	one dot on the map represents n (in var units).

iter	number of iteration to try to locate sample points (see Details).
pch	symbol to use: points.
cex	size of the symbols
type	points allocation method: "random" or "regular" (see Details).
col	color of the points.
legend.pos	"topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.
legend.txt	text in the legend.
legend.cex	size of the legend text.
legend.col	color of the text in the legend.
legend.frame	whether to add a frame to the legend (TRUE) or not (FALSE).
add	whether to add the layer to an existing plot (TRUE) or not (FALSE).

Details

The iter parameter is defined within the spsample function. If an error occurred, increase this value. The type parameters is defined within the spsample function.

See Also

propSymbolsLayer

Examples

```
data("nuts2006")
# Example 1
plot(nuts0.spdf)
dotDensityLayer(spdf = nuts0.spdf, df=nuts0.df,var="pop2008")


# Example 2
layoutLayer(title = "Population in Europe, 2008",
            sources = "Eurostat, 2008",
            scale = NULL,
            frame = TRUE,
            col = "black",
            coltitle = "white",
            bg = "#E6E6E6",
            extent = nuts0.spdf)
plot(nuts1.spdf, col = "#B8704D50",border = "white", add=TRUE)
dotDensityLayer(spdf = nuts1.spdf, df=nuts1.df,var="pop2008",
                type = "regular", pch=20, col = "brown",
                n = 100000)
```

frame.spdf *Frame around Europe*

Description

Frame around European countries.

Format

SpatialPolygonsDataFrame.

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

getBorders *Extract SpatialPolygonsDataFrame Borders*

Description

Extract borders between SpatialPolygonsDataFrame units.

Usage

```
getBorders(spdf, spdfid = NULL)
```

Arguments

spdf	a SpatialPolygonsDataFrame. This SpatialPolygonsDataFrame has to be projected (planar coordinates).
spdfid	identifier field in spdf, default to the first column of the spdf data frame. (optional)

Value

A SpatialLinesDataFrame of borders is returned. This object has three id fields: id, id1 and id2. id1 and id2 are ids of units that neighbour a border; id is the concatenation of id1 and id2 (with "_" as separator).

Note

getBorders and getOuterBorders can be combined with rbind.

See Also

discLayer, getOuterBorders

Examples

```
data(nuts2006)
# Get units borders
nuts0.contig.spdf <- getBorders(nuts0.spdf)
# Random colors
nuts0.contig.spdf$col <-
# Plot Countries
plot(nuts0.spdf, border = NA, col = "grey60")
# Plot borders
plot(nuts0.contig.spdf, col =
       sample(x = rainbow(nrow(nuts0.contig.spdf))),
     lwd = 3, add = TRUE)
```

`getBreaks` *Discretization*

Description

A function to discretize continious variables.

Usage

```
getBreaks(v, nclass = NULL, method = "quantile", k = 1, middle = FALSE)
```

Arguments

`v`	a vector of numeric values.
`nclass`	a number of classes
`method`	a discretization method; one of "sd", "equal", "quantile", "fisher-jenks","q6", "geom" or "msd" (see Details).
`k`	number of standard deviation for "msd" method (see Details)..
`middle`	creation of a central class for "msd" method (see Details).

Details

"sd", "equal", "quantile" and "fisher-jenks" are classIntervals methods.

Jenks and Fisher-Jenks algorithms are based on the same principle and give quite similar results but Fisher-Jenks is much faster.

The "q6" method uses the following quantile probabilities: 0, 0.05, 0.275, 0.5, 0.725, 0.95, 1.

The "geom" method is based on a geometric progression along the variable values.

The "msd" method is based on the mean and the standard deviation of a numeric vector. The nclass parameter is not relevant, use k and middle instead. k indicates the extent of each class in share of standard deviation. If middle=TRUE then the mean value is the center of a class else the mean is a break value.

Value

A numeric vector of breaks

Note

This function is mainly a wrapper around classIntervals function of the classInt package + q6, geom and msd methods.

Examples

```
data("nuts2006")
# Create the natality rate
var <- nuts2.df$birth_2008/nuts2.df$pop2008 * 1000

# Histogram
hist(var, probability = TRUE, nclass = 30)
rug(var)
moy <- mean(var)
med <- median(var)
abline(v = moy, col = "red", lwd = 3)
abline(v = med, col = "blue", lwd = 3)

# Quantile intervals
breaks <- getBreaks(v = var, nclass = 6, method = "quantile")
hist(var, probability = TRUE, breaks = breaks, col = "#F0D9F9")
rug(var)
med <- median(var)
abline(v = med, col = "blue", lwd = 3)

# Geometric intervals
breaks <- getBreaks(v = var, nclass = 8, method = "geom")
hist(var, probability = TRUE, breaks = breaks, col = "#F0D9F9")
rug(var)

# Mean and standard deviation (msd)
breaks <- getBreaks(v = var, method = "msd", k = 1, middle = TRUE)
hist(var, probability = TRUE, breaks = breaks, col = "#F0D9F9")
rug(var)
moy <- mean(var)
sd <- sd(var)
abline(v = moy, col = "red", lwd = 3)
abline(v = moy + 0.5 * sd, col = "blue", lwd = 3)
abline(v = moy - 0.5 * sd, col = "blue", lwd = 3)
```

---

getFigDim *Get Figure Dimensions*

Description

Give the dimension of a map figure to be exported in raster of vector format.
Output dimension are based on a Spatial*DataFrame dimension ratio, the margins of the figure, a targeted width or height of the figure and its resolution.

Usage

```
getFigDim(spdf, width = NULL, height = NULL, mar = par("mar"), res = 72)
```

Arguments

spdf	a Spatial*DataFrame.
width	width of the figure (in pixels), either width or height must be set.
height	heigth of the figure (in pixels), either width or height must be set.
mar	a numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot (see par).
res	the nominal resolution in ppi which will be recorded in the bitmap file.

Details

The function can be used to export vector or raster files (see examples).

Value

A vector of width and height in pixels is returned.

Examples

```
## Not run:
data("nuts2006")
spdf <- nuts0.spdf[nuts0.spdf$id=="IT",]

## PNG export
# get figure dimension
sizes <- getFigDim(spdf = spdf, width = 450, mar = c(0,0,1.2,0))
# export the map
png(filename = "Italy.png", width = sizes[1], height = sizes[2])
par(mar = c(0,0,1.2,0))
plot(spdf, col = NA, border=NA, bg = "#A6CAE0")
plot(world.spdf, col = "#E3DEBF", border = NA, add = TRUE)
plot(spdf, col = "#D1914D", border = "white", add = TRUE)
layoutLayer(title = "Map of Italy")
dev.off()

## PDF export
# get figure dimension
sizes <- getFigDim(spdf = spdf, width = 450, mar = c(1,1,2.2,1))
# export the map
pdf(file = "Italy.pdf", width = sizes[1]/72, height = sizes[2]/72)
par(mar = c(1,1,2.2,1))
```

```
plot(spdf, col = NA, border = NA, bg = "#A6CAE0")
plot(world.spdf, col = "#E3DEBF", border = NA, add = TRUE)
plot(spdf, col = "#D1914D", border = "white", add = TRUE)
layoutLayer(title = "Map of Italy")
dev.off()

## End(Not run)
```

`getGridData` *Compute Data for a Grid Layer*

Description

This function computes data to match a grid layer (as outputed by getGridLayer) according to the surface intersections.

Usage

```
getGridData(x, df, dfid = NULL, var)
```

Arguments

x	a list generated by the getGridLayer function.
df	a data frame that contains the values to adapt to the grid. It must correspond to the spdf argument in getGridLayer.
dfid	identifier field in df, default to the first column of df. (optional)
var	name of the numeric field in df to adapt to the grid.

Value

A data frame is returned. id_cell are ids of the grid, the two other variable are the share of the variable in each cell and the share of the variable in each cell divided by its area (in map units).

See Also

getGridLayer

Examples

```
## Not run:
data(nuts2006)
# Create a grid layer
mygrid <- getGridLayer(spdf=nuts2.spdf,cellsize = 200000)
# Compute data for the grid layer
datagrid.df <- getGridData(mygrid, nuts2.df, "pop2008",dfid=NULL)

# Plot total population
plot(mygrid$spdf, col="#CCCCCC",border="white")
```

```
propSymbolsLayer(spdf = mygrid$spdf, df = datagrid.df, legend.style = "e",
             legend.pos = "right", border = "white",legend.title.txt = "Total population",
                var = "pop2008", k=0.005, col="black", add=TRUE)

# Plot dentsity of population
## conversion from square meter to square kilometers
datagrid.df$densitykm <- datagrid.df$pop2008_density*1000*1000
cols <- carto.pal(pal1 = "taupe.pal", n1 = 6)
choroLayer(spdf = mygrid$spdf,df = datagrid.df,var = "densitykm", add=FALSE,
           border = "grey80",col=cols,
           legend.pos = "right", method = "q6",
           legend.title.txt = "Population density")

## End(Not run)
```

`getGridLayer` *Build a Regular Grid Layer*

Description

Build a regular grid based on a SpatialPolygonsDataFrame. Provide also a table of surface intersections.

Usage

```
getGridLayer(spdf, cellsize, spdfid = NULL)
```

Arguments

`spdf`	a SpatialPolygonsDataFrame.
`cellsize`	output cell size, in map units.
`spdfid`	identifier field in spdf, default to the first column of the spdf data frame. (optional)

Value

A list is returned. The list contains "spdf": a SpatialPolygonsDataFrame of a regular grid and "df": a data frame of surface intersection. df fields are id_cell: ids of the grid; id_geo: ids of the spdf and area_pct: share of the area of the polygon in the cell (a value of 55 means that 55% of the spdf unit is within the cell).

See Also

getGridData

Examples

```
## Not run:
data(nuts2006)
# Get a grid layer
mygrid <- getGridLayer(spdf = nuts2.spdf, cellsize = 200000)
# Plot the grid
plot(mygrid$spdf)
head(mygrid$df)

## End(Not run)
```

`getLinkLayer` *Create a SpatialLinesDataFrame from a Data Frame of Links.*

Description

Create a SpatialLinesDataFrame from a data frame of links.

Usage

```
getLinkLayer(spdf, spdf2 = NULL, df, spdfid = NULL, spdf2id = NULL,
  dfids = NULL, dfide = NULL)
```

Arguments

`spdf`	a SpatialPointsDataFrame or a SpatialPolygonsDataFrame; layer used to get starting points of links. If spdf2 is NULL, spdf is also used to get ending points. If spdf is a SpatialPolygonsDataFrame, links start (or end) at centroids.
`spdf2`	a SpatialPointsDataFrame or a SpatialPolygonsDataFrame; layer used to get ending points of links. If spdf2 is a SpatialPolygonsDataFrame, links start (or end) at centroids. (optional)
`df`	a data frame that contains identifiers of starting and ending points.
`spdfid`	identifier field in spdf, default to the first column of the spdf data frame. (optional)
`spdf2id`	identifier field in spdf2, default to the first column of the spdf2 data frame. (optional)
`dfids`	identifier field of starting points of links in df, default to the first column of df. (optional)
`dfide`	identifier field of ending points of links in df, default to the second column of df. (optional)

Value

A SpatialLinesDataFrame is returned, its data frame contains two fields (dfids and dfide).

See Also

gradLinkLayer, propLinkLayer

Examples

```
data("nuts2006")
# Create a link layer
head(twincities)
# Select links from Ireland (IE)
twincitiesIE <- twincities[substr(twincities$i,1,2)=="IE", ]
twincities.spdf <- getLinkLayer(spdf = nuts2.spdf, df = twincitiesIE[,1:2])
# Plot the links
plot(nuts2.spdf, col = "#6C6870")
plot(twincities.spdf, col = "#F78194", add = TRUE)
```

---

getOuterBorders *Extract SpatialPolygonsDataFrame Outer Borders*

Description

Extract outer borders between SpatialPolygonsDataFrame units. Outer borders are non-contiguous SpatialPolygonsDataFrame borders (e.g. maritim borders).

Usage

```
getOuterBorders(spdf, spdfid = NULL, res = NULL, width = NULL)
```

Arguments

spdf	a SpatialPolygonsDataFrame.
spdfid	identifier field in spdf, default to the first column of the spdf data frame. (optional)
res	resolution of the grid used to compute borders (in spdf units). A high resolution will give more detailed borders. (optional)
width	maximum distance between used to compute borders (in spdf units). A higher width will build borders between units that are farther apart. (optional)

Value

A SpatialLinesDataFrame of borders is returned. This object has three id fields: id, id1 and id2. id1 and id2 are ids of units that neighbour a border; id is the concatenation of id1 and id2 (with "_" as separator).

Note

getBorders and getOuterBorders can be combined with rbind.

See Also

discLayer, getBorders

Examples

```
## Not run:
data(nuts2006)
# Get units borders
nuts0.outer <- getOuterBorders(nuts0.spdf)
# Plot Countries
plot(nuts0.spdf, border = NA, col = "grey60")
# Plot borders
plot(nuts0.outer, col = sample(x = rainbow(nrow(nuts0.outer))),
     lwd = 3, add = TRUE)

## End(Not run)
```

---

getTiles *Get Tiles from Open Map Servers*

Description

Get map tiles based on a Spatial*DataFrame extent. Maps can be fetched from various open map servers.

Usage

```
getTiles(spdf, type = "osm", zoom = NULL, crop = FALSE)
```

Arguments

spdf	a Spatial*DataFrame with a valid projection attribute.
type	the tile server from which to get the map, one of "hikebike", "hotstyle", "lovinacycle", "lovinahike", "mapquestosm", "mapquestsat", "opencycle", "openpiste", "osm", "osmgrayscale", "osmtransport", "stamenbw", "stamenwatercolor", "thunderforestlandscape" and "thunderforestoutdoors".
zoom	the zoom level. If null, it is determined automatically (see Details).
crop	TRUE if results should be cropped to the specified spdf extent, FALSE otherwise.

Details

Zoom levels are descibed on the OpenStreetMap wiki: http://wiki.openstreetmap.org/wiki/Zoom_levels.

Value

A RatserBrick is returned.

Note

This function is a wrapper around the `osm.raster` function from the `rosm` package.
Use directly the rosm package to have a finer control over extraction and display parameters.

See Also

tilesLayer

Examples

```
## Not run:
data("nuts2006")
# extract Denmark
spdf <- nuts0.spdf[nuts0.spdf$id=="DK",]
# Download the tiles, extent = Denmark
den <- getTiles(spdf = spdf, type = "osm", crop = TRUE)
class(den)
# Plot the tiles
tilesLayer(den)
# Plot countries
plot(spdf, add=TRUE)
# Map tiles sources
mtext(text = "Map data © OpenStreetMap contributors, under CC BY SA.",
      side = 1, adj = 0, cex = 0.7, font = 3)

## End(Not run)
```

---

gradLinkLayer *Graduated Links Layer*

Description

Plot a layer of graduated links. Links are plotted according to discrete classes of widths.

Usage

```
gradLinkLayer(spdf, df, spdfid = NULL, spdfids, spdfide, dfid = NULL, dfids,
  dfide, var, breaks = getBreaks(v = df[, var], nclass = 4, method =
  "quantile"), lwd = c(1, 2, 4, 6), col = "red",
  legend.pos = "bottomleft", legend.title.txt = var,
  legend.title.cex = 0.8, legend.values.cex = 0.6, legend.values.rnd = 0,
  legend.frame = FALSE, add = TRUE)
```

Arguments

spdf	SpatialLinesDataFrame; a link layer.
df	data frame with identifier(s) and a variable.
spdfid	unique identifier in spdf (spdfids, spdfide, dfids and dfide are not used).

spdfids	identifier of starting points in spdf (spdfid and dfid are not used).
spdfide	identifier of ending points in spdf (spdfid and dfid are not used).
dfid	unique identifier in df (spdfids, spdfide, dfids and dfide are not used).
dfids	identifier of starting points in df (spdfid and dfid are not used).
dfide	identifier of ending points in df (spdfid and dfid are not used).
var	name of the variable used to plot the links widths.
breaks	break values in sorted order to indicate the intervals for assigning the lines widths.
lwd	vector of widths (classes of widths).
col	color of the links.
legend.pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.
legend.title.txt	title of the legend.
legend.title.cex	size of the legend title.
legend.values.cex	size of the values in the legend.
legend.values.rnd	number of decimal places of the values displayed in the legend.
legend.frame	whether to add a frame to the legend (TRUE) or not (FALSE).
add	whether to add the layer to an existing plot (TRUE) or not (FALSE).

Note

Unlike most of cartography functions, identifiers fields are mandatory.

See Also

getLinkLayer, propLinkLayer, legendGradLines

Examples

```
data("nuts2006")
# Create a link layer
twincities.spdf <- getLinkLayer(spdf = nuts2.spdf, df = twincities[,1:2])
# Plot the links - Twin cities agreements between regions
plot(nuts0.spdf, col = "grey60",border = "grey20")
gradLinkLayer(spdf = twincities.spdf, df = twincities,
              spdfids = "i", spdfide = "j",
              dfids = "i", dfide = "j",legend.pos = "topright",
              var = "fij", breaks = c(2,5,15,20,30), lwd = c(0.1,1,4,10),
              col = "#92000090", add = TRUE)
```

gradLinkTypoLayer *Graduated and Colored Links Layer*

Description

Plot a layer of colored and graduated links. Links are plotted according to discrete classes of widths. Colors depends on a discrete variable of categories.

Usage

```
gradLinkTypoLayer(spdf, df, spdfid = NULL, spdfids, spdfide, dfid = NULL,
  dfids, dfide, var, breaks = getBreaks(v = df[, var], nclass = 4, method =
  "quantile"), lwd = c(1, 2, 4, 6), var2, col = NULL, colNA = "white",
  legend.title.cex = 0.8, legend.values.cex = 0.6, legend.values.rnd = 0,
  legend.var.pos = "bottomleft", legend.var.title.txt = var,
  legend.var.frame = FALSE, legend.var2.pos = "topright",
  legend.var2.title.txt = var2, legend.var2.values.order = NULL,
  legend.var2.nodata = "no data", legend.var2.frame = FALSE, add = TRUE)
```

Arguments

spdf	SpatialLinesDataFrame; a link layer.
df	data frame with identifier(s) and a variable.
spdfid	unique identifier in spdf (spdfids, spdfide, dfids and dfide are not used).
spdfids	identifier of starting points in spdf (spdfid and dfid are not used).
spdfide	identifier of ending points in spdf (spdfid and dfid are not used).
dfid	unique identifier in df (spdfids, spdfide, dfids and dfide are not used).
dfids	identifier of starting points in df (spdfid and dfid are not used).
dfide	identifier of ending points in df (spdfid and dfid are not used).
var	name of the variable used to plot the links widths.
breaks	break values in sorted order to indicate the intervals for assigning the lines widths.
lwd	vector of widths (classes of widths).
var2	name of the variable used to plot the links colors.
col	color of the links.
colNA	no data color.
legend.title.cex	size of the legend title.
legend.values.cex	size of the values in the legend.
legend.values.rnd	number of decimal places of the values in the legend.

`legend.var.pos` position of the legend for var, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".

`legend.var.title.txt` title of the legend (numeric data).

`legend.var.frame` whether to add a frame to the legend (TRUE) or not (FALSE).

`legend.var2.pos` position of the legend for var2, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".

`legend.var2.title.txt` title of the legend (factor data).

`legend.var2.values.order` values order in the legend, a character vector that matches var modalities. Colors will be affected following this order.

`legend.var2.nodata` text for "no data" values

`legend.var2.frame` whether to add a frame to the legend (TRUE) or not (FALSE).

`add` whether to add the layer to an existing plot (TRUE) or not (FALSE).

Note

Unlike most of cartography functions, identifiers fields are mandatory.

See Also

getLinkLayer, propLinkLayer, legendGradLines, gradLinkLayer

Examples

```
data("nuts2006")
# Create a link layer
twincities.spdf <- getLinkLayer(spdf = nuts2.spdf, df = twincities[,1:2])

# Plot the links - Twin cities agreements between regions
plot(nuts0.spdf, col = "grey60",border = "grey20")

# Countries of agreements
twincities$ctry <- substr(twincities$j,1,2)

# Agreements with german cities
twincitiesok <- twincities[substr(twincities$i,1,2)=="DE",]

# plot the colored and graduated links
gradLinkTypoLayer(spdf = twincities.spdf, df = twincitiesok,
                  spdfids = "i", spdfide = "j",
                  dfids = "i", dfide = "j",
                  var = "fij", breaks = c(5,10,15,20),
                  lwd = c(1,4,8),
                  var2 = "ctry",  add = TRUE)
```

graticule.spdf *Graticule around Europe*

Description

Graticule around Europe.

Format

SpatialLines.

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

labelLayer *Label Layer*

Description

Put labels on a map.

Usage

```
labelLayer(spdf, df, spdfid = NULL, dfid = NULL, txt, col = "black",
  cex = 0.7, ...)
```

Arguments

spdf	a SpatialPointsDataFrame or a SpatialPolygonsDataFrame; if spdf is a SpatialPolygonsDataFrame texts are plotted on centroids.
df	a data frame that contains the labels to plot. If df is missing spdf@data is used instead.
spdfid	identifier field in spdf, default to the first column of the spdf data frame. (optional)
dfid	identifier field in df, default to the first column of df. (optional)
txt	labels field in df.
col	labels color.
cex	labels cex.
...	further text arguments, such as pos or adj.

See Also

layoutLayer

Examples

```
data("nuts2006")

# Layout plot
layoutLayer(  title = "Most Populated Countries of Europe",
              author = "", sources = "",
              scale = NULL,col = NA, coltitle = "black",
              frame = FALSE, bg = "#A6CAE0",
              south = TRUE, extent = nuts0.spdf)

plot(world.spdf, col  = "#E3DEBF", border=NA, add=TRUE)
plot(nuts0.spdf, col = "#D1914D",border = "white", lwd=1, add=TRUE)

# Selection of the 10 most populated countries of Europe
dflab <- nuts0.df[order(nuts0.df$pop2008, decreasing = TRUE),][1:10,]

# Label creation
dflab$lab <- paste(dflab$id, "\n", round(dflab$pop2008/1000000,0), "M", sep ="")

# Label plot of the 10 most populated countries
labelLayer(spdf = nuts0.spdf, df = dflab, txt = "lab",
           col = "#690409", cex = 0.9, font = 2)
text(x = 5477360, y = 4177311, labels = "The 10 most populated countries of Europe
Total population 2008, in millions of inhabitants.",
     cex = 0.7, adj = 0)
```

---

layoutLayer *Layout Layer*

Description

Plot a layout layer.

Usage

```
layoutLayer(title = "Title of the map, year", sources = "Source(s)",
  author = "Author(s)", col = "black", coltitle = "white", theme = NULL,
  bg = NULL, scale = 0, frame = TRUE, north = FALSE, south = FALSE,
  extent = NULL)
```

Arguments

title	title of the map.
sources	sources of the map (or something else).
author	author of the map (or something else).
col	color of the title box and frame border.
coltitle	color of the title.

theme	name of a cartographic palette (see carto.pal.info). col and coltitle are set according to the chosen palette.
bg	color of the frame background.
scale	size of the scale in kilometers. If set to NULL, no scale is displayed, if set to 0 an automatic scale is displayed (1/10 of the map width).
frame	wheither displaying a frame (TRUE) or not (FALSE).
north	wheither displaying a Noth arrow (TRUE) or not (FALSE).
south	wheither displaying a South arrow (TRUE) or not (FALSE).
extent	a SpatialPolygonsDataFrame or a SpatialPointsDataFrame; set the extent of the frame to the one of a Spatial object. (optional)

Details

If extent is not set, plot.new has to be called first.
The size of the title box in layoutLayer is fixed to 1.2 lines height.

See Also

labelLayer

Examples

```
data("nuts2006")
# Example 1
plot(nuts0.spdf, col = "grey60",border = "grey20", add=FALSE)
# Layout plot
layoutLayer()

# Example 2
plot(nuts0.spdf, col=NA, border = NA, bg ="#A6CAE0")
plot(world.spdf, col  = "#E3DEBF", border=NA, add=TRUE)
plot(nuts0.spdf, col = "#D1914D",border = "white", lwd=1, add=TRUE)
layoutLayer(col = NA, coltitle = "black",
            sources = "", author = "",
            frame = FALSE,
            south = TRUE)

# Example 3
nuts3.df$gdphab <- 1000000 * nuts3.df$gdppps2008 / nuts3.df$pop2008
choroLayer(spdf = nuts3.spdf, df = nuts3.df, var = "gdphab",
           legend.pos = "right", border = NA, nclass = 6,
           col = carto.pal('green.pal', 6))
# Layout plot
layoutLayer(title = "GDP per Inhabitants", sources = "",
            author = "Eurostat, 2008", theme = "green.pal")
```

`legendBarsSymbols` *Legend for Proportional Bars Maps*

Description

Plot legend for proportional bars maps

Usage

```
legendBarsSymbols(pos = "topleft", title.txt = "Title of the legend",
  title.cex = 0.8, cex = 1, values.cex = 0.6, var, r, breakval = NULL,
  col = "red", col2 = "blue", frame = FALSE, values.rnd = 0,
  style = "c")
```

Arguments

`pos`	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".
`title.txt`	title of the legend.
`title.cex`	size of the legend title.
`cex`	size of the legend. 2 means two times bigger.
`values.cex`	size of the values in the legend.
`var`	vector of values.
`r`	a vector giving the heights of the bars.
`breakval`	breaking value (see Details).
`col`	color of symbols.
`col2`	second color of symbols (see Details).
`frame`	whether to add a frame to the legend (TRUE) or not (FALSE).
`values.rnd`	number of decimal places of the values in the legend.
`style`	either "c" or "e". The legend has two display styles, "c" stands for compact and "e" for extended.

Details

The breakval parameter allows to plot symbols of two colors: the first color (col) for values superior or equal to breakval, second color (col2) for values inferior to breakval.

Examples

```
data("nuts2006")
plot(nuts0.spdf)

legendBarsSymbols(pos = "topleft", title.txt = "Title of\nthe legend",
                    title.cex = 0.8, values.cex = 0.6,cex = 3,
                    var = nuts1.df$pop2008,
                    r = sqrt((abs(nuts1.df$pop2008) * 100000) / pi),
                    col = "purple",
                    values.rnd=0, style ="e")
```

`legendChoro` *Legend for Choropleth Maps*

Description

Plot legend for choropleth maps.

Usage

```
legendChoro(pos = "topleft", title.txt = "Title of the legend",
  title.cex = 0.8, values.cex = 0.6, breaks, col, cex = 1,
  values.rnd = 2, nodata = TRUE, nodata.txt = "No data",
  nodata.col = "white", frame = FALSE, symbol = "box")
```

Arguments

`pos`	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".
`title.txt`	title of the legend.
`title.cex`	size of the legend title.
`values.cex`	size of the values in the legend.
`breaks`	break points in sorted order to indicate the intervals for assigning the colors. Note that if there are nlevel colors (classes) there should be (nlevel+1) breakpoints.
`col`	a vector of colors.
`cex`	size of the legend. 2 means two times bigger.
`values.rnd`	number of decimal places of the values in the legend.
`nodata`	if TRUE a "no data" box or line is plotted.
`nodata.txt`	label for "no data" values.
`nodata.col`	color of "no data" values.
`frame`	whether to add a frame to the legend (TRUE) or not (FALSE).
`symbol`	type of symbol in the legend 'line' or 'box'

Examples

```
data("nuts2006")
plot(nuts0.spdf, col = "grey")
box()
legendChoro(pos = "bottomleft", title.txt = "Title of the legend", title.cex = 0.8,
            values.cex = 0.6, breaks = c(1,2,3,4,10.27,15.2),
            col = carto.pal(pal1 = "orange.pal",n1 = 5), values.rnd =2,
            nodata = TRUE, nodata.txt = "No data available", frame = TRUE, symbol="box")
legendChoro(pos = "bottomright", title.txt = "Title of the legend", title.cex = 0.8,
            values.cex = 0.6, breaks = c(1,2,5,7,10,15.27),
            col = carto.pal(pal1 = "wine.pal",n1 = 5), values.rnd = 0,
            nodata = TRUE, nodata.txt = "NA",nodata.col = "black",
            frame = TRUE, symbol="line")
```

---

legendCirclesSymbols *Legend for Proportional Circles Maps*

Description

Plot legend for proportional circles maps

Usage

```
legendCirclesSymbols(pos = "topleft", title.txt = "Title of the legend",
  title.cex = 0.8, cex = 1, values.cex = 0.6, var, r, breakval = NULL,
  col = "red", col2 = "blue", frame = FALSE, values.rnd = 0,
  style = "c")
```

Arguments

pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".
title.txt	title of the legend.
title.cex	size of the legend title.
cex	size of the legend. 2 means two times bigger.
values.cex	size of the values in the legend.
var	vector of values.
r	a vector giving the radii of the circles.
breakval	breaking value (see Details).
col	color of symbols.
col2	second color of symbols (see Details).
frame	whether to add a frame to the legend (TRUE) or not (FALSE).
values.rnd	number of decimal places of the values in the legend.
style	either "c" or "e". The legend has two display styles, "c" stands for compact and "e" for extended.

Details

The breakval parameter allows to plot symbols of two colors: the first color (col) for values superior or equal to breakval, second color (col2) for values inferior to breakval.

Examples

```
data("nuts2006")
plot(nuts0.spdf)

legendCirclesSymbols(pos = "topleft", title.txt = "Title of\nthe legend",
                     title.cex = 0.8, values.cex = 0.6,cex = 1.5,
                     var = nuts1.df$pop2008,
                     r = sqrt((abs(nuts1.df$pop2008) * 100000) / pi),
                     col = "pink", frame = TRUE,
                     values.rnd=0, style ="c")
```

---

legendGradLines *Legend for Graduated Size Lines Maps*

Description

Plot legend for graduated size lines maps.

Usage

```
legendGradLines(pos = "topleft", title.txt = "Title of the legend",
  title.cex = 0.8, values.cex = 0.6, breaks, lwd, col, values.rnd = 2,
  cex = 1, frame = FALSE)
```

Arguments

pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".
title.txt	title of the legend.
title.cex	size of the legend title.
values.cex	size of the values in the legend.
breaks	break points in sorted order to indicate the intervals for assigning the width of the lines
lwd	a vector giving the width of the lines.
col	color of symbols.
values.rnd	number of decimal places of the values in the legend.
cex	size of the legend. 2 means two times bigger.
frame	whether to add a frame to the legend (TRUE) or not (FALSE).

Examples

```
data("nuts2006")
plot(nuts0.spdf)
box()
legendGradLines(title.txt = "Title of the legend",
                pos = "topright",
                title.cex = 0.8,
                values.cex = 0.6, breaks = c(1,2,3,4,10.2,15.2),
                lwd = c(0.2,2,4,5,10),
                col ="blue", values.rnd =2)
```

---

legendPropLines *Legend for Proportional Lines Maps*

Description

Plot legend for proportional lines maps

Usage

```
legendPropLines(pos = "topleft", title.txt = "Title of the legend",
  title.cex = 0.8, cex = 1, values.cex = 0.6, var, lwd, col = "red",
  frame = FALSE, values.rnd = 0)
```

Arguments

pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".
title.txt	title of the legend.
title.cex	size of the legend title.
cex	size of the legend. 2 means two times bigger.
values.cex	size of the values in the legend.
var	vector of values.
lwd	a vector giving the width of the lines.
col	color of symbols.
frame	whether to add a frame to the legend (TRUE) or not (FALSE).
values.rnd	number of decimal places of the values in the legend.

Examples

```
data("nuts2006")
plot(nuts0.spdf)
box()
legendPropLines(pos = "topleft", title.txt = "Title",
                title.cex = 0.8, values.cex = 0.6, cex = 2,
                var = nuts1.df$pop2008,
                lwd = nuts1.df$pop2008/1000000,
                col="red", frame=TRUE, values.rnd=2)
```

`legendPropTriangles` *Legend for Double Proportional Triangles Maps*

Description

Plot legends for double proportional triangles maps.

Usage

```
legendPropTriangles(pos = "topleft", title.txt, var.txt, var2.txt,
  title.cex = 0.8, cex = 1, values.cex = 0.6, var, var2, r, r2,
  col = "red", col2 = "blue", frame = FALSE, values.rnd = 0,
  style = "c")
```

Arguments

`pos`	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".
`title.txt`	title of the legend.
`var.txt`	name of var.
`var2.txt`	name of var2.
`title.cex`	size of the legend title.
`cex`	size of the legend. 2 means two times bigger.
`values.cex`	size of the values in the legend.
`var`	a first vector of positive values.
`var2`	a second vector of positive values.
`r`	a first vector of sizes.
`r2`	a second vector of sizes.
`col`	color of symbols.
`col2`	second color of symbols.
`frame`	whether to add a frame to the legend (TRUE) or not (FALSE).
`values.rnd`	number of decimal places of the values in the legend.
`style`	either "c" or "e". The legend has two display styles, "c" stands for compact and "e" for extended.

Examples

```
data("nuts2006")
plot(nuts0.spdf)
box()
var <- round((nuts0.df$pop2008 / sum(nuts0.df$pop2008))*100,2)
var2 <- round((nuts0.df$gdppps2008 / sum(nuts0.df$gdppps2008))*100,2)
r <- sqrt(var)/2*1000000
```

```
r2 <- sqrt(var2)/2*1000000
legendPropTriangles(pos = "topright", var.txt = "population totale (habs)",
                        var2.txt = "pib (euros)", title.txt="PIB par habitant",
                        title.cex = 0.8, values.cex = 0.6, cex = 1,
                        var = var, var2 = var2, r = r, r2 = r2,
                        col="green", col2="yellow", frame=TRUE, values.rnd=2,
                        style="c")
```

`legendSquaresSymbols` *Legend for Proportional Squares Maps*

Description

Plot legend for proportional squares maps

Usage

```
legendSquaresSymbols(pos = "topleft", title.txt = "Title of the legend",
  title.cex = 0.8, cex = 1, values.cex = 0.6, var, r, breakval = NULL,
  col = "red", col2 = "blue", frame = FALSE, values.rnd = 0,
  style = "c")
```

Arguments

`pos`	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".
`title.txt`	title of the legend.
`title.cex`	size of the legend title.
`cex`	size of the legend. 2 means two times bigger.
`values.cex`	size of the values in the legend.
`var`	vector of values.
`r`	a vector giving the length of the sides of the squares.
`breakval`	breaking value (see Details).
`col`	color of symbols.
`col2`	second color of symbols (see Details).
`frame`	whether to add a frame to the legend (TRUE) or not (FALSE).
`values.rnd`	number of decimal places of the values in the legend.
`style`	either "c" or "e". The legend has two display styles, "c" stands for compact and "e" for extended.

Details

The breakval parameter allows to plot symbols of two colors: the first color (col) for values superior or equal to breakval, second color (col2) for values inferior to breakval.

Examples

```
data("nuts2006")
plot(nuts0.spdf)
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4], border = "black")
legendSquaresSymbols(pos = "bottomright", title.txt = "Title of\nthe legend ",
                     title.cex = 0.8, values.cex = 0.6,
                     var = nuts1.df$pop2008,
                     r = sqrt((abs(nuts1.df$pop2008) * 5000)),
                     breakval=10, col="red", col2="blue",
                     frame=TRUE, values.rnd=0, style ="c")
```

`legendTypo` *Legend for Typology Maps*

Description

Plot legend for typology maps.

Usage

```
legendTypo(pos = "topleft", title.txt = "Title of the legend",
  title.cex = 0.8, values.cex = 0.6, col, categ, cex = 1, nodata = TRUE,
  nodata.txt = "No data", nodata.col = "white", frame = FALSE,
  symbol = "box")
```

Arguments

`pos`	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".
`title.txt`	title of the legend.
`title.cex`	size of the legend title.
`values.cex`	size of the values in the legend.
`col`	a vector of colors.
`categ`	vector of categories.
`cex`	size of the legend. 2 means two times bigger.
`nodata`	if TRUE a "no data" box or line is plotted.
`nodata.txt`	label for "no data" values.
`nodata.col`	color of "no data" values.
`frame`	whether to add a frame to the legend (TRUE) or not (FALSE).
`symbol`	character; 'line' or 'box'

Examples

```
data("nuts2006")
plot(nuts0.spdf, col = "grey")
box()

# Define labels and colors
someLabels <- c("red color", "yellow color", "green color", "black color")
someColors <- c("red", "yellow", "green", "black")

# plot legend
legendTypo(pos = "bottomleft", title.txt = "Title of the legend", title.cex = 0.8,
           values.cex = 0.6, col = someColors, categ = someLabels,
           cex = 0.75,
           nodata = TRUE, nodata.txt = "no data", frame = TRUE, symbol="box")
legendTypo(pos = "topright", title.txt = "",
           title.cex = 1.5, cex = 1.25,
           values.cex = 1, col = someColors, categ = someLabels,
           nodata = FALSE, frame = FALSE, symbol="line")
```

---

nuts0.df *Nuts0 Dataset*

Description

This dataset contains some socio-economic data

Details

This data frame can be used with the SpatialPolygonsDataFrame nuts0.spdf

Fields

`id` Unique nuts id (character)

`emp2008` Active population in employment in 2008 (thousands persons) (numeric)

`act2008` Active population in 2008 (thousands persons) (numeric)

`unemp2008` Active population unemployed in 2008 (thousands persons) (numeric)

`birth_2008` Number of birth in 2008 (live birth) (numeric)

`death_2008` Number of death in 2008 (death) (numeric)

`gdppps1999` Gross domestic product (Purchasing Power Standards) in 1999 (million euros) (numeric)

`gdppps2008` Gross domestic product (Purchasing Power Standards) in 2008 (million euros) (numeric)

`pop1999` Total population in 1999 (inhabitants) (numeric)

`pop2008` Total population in 2008 (inhabitants) (numeric)

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=151

nuts0.spdf *Nuts0 Regions*

Description

Delineations of EU administrative units (level 0, 2006 version).

Format

SpatialPolygonsDataFrame.

Details

This SpatialPolygonsDataFrame can be used with the nuts0.df data frame

Fields

`id` Unique nuts id (character)

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

nuts1.df *Nuts1 Dataset*

Description

This dataset contains some socio-economic data

Details

This data frame can be used with the SpatialPolygonsDataFrame nuts1.spdf

Fields

`id` Unique nuts id (character)

`emp2008` Active population in employment in 2008 (thousands persons) (numeric)

`act2008` Active population in 2008 (thousands persons) (numeric)

`unemp2008` Active population unemployed in 2008 (thousands persons) (numeric)

`birth_2008` Number of birth in 2008 (live birth) (numeric)

`death_2008` Number of death in 2008 (death) (numeric)

`gdppps1999` Gross domestic product (Purchasing Power Standards) in 1999 (million euros) (numeric)

gdppps2008 Gross domestic product (Purchasing Power Standards) in 2008 (million euros) (numeric)

pop1999 Total population in 1999 (inhabitants) (numeric)

pop2008 Total population in 2008 (inhabitants) (numeric)

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=151

---

nuts1.spdf *Nuts1 Regions*

Description

Delineations of EU administrative units (level 1, 2006 version).

Format

SpatialPolygonsDataFrame.

Details

This SpatialPolygonsDataFrame can be used with the nuts1.df data frame

Fields

id Unique nuts id (character)

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

---

nuts2.df *Nuts2 Dataset*

Description

This dataset contains some socio-economic data

Details

This data frame can be used with the SpatialPolygonsDataFrame nuts2.spdf

Fields

`id` Unique nuts id (character)

`emp2008` Active population in employment in 2008 (thousands persons) (numeric)

`act2008` Active population in 2008 (thousands persons) (numeric)

`unemp2008` Active population unemployed in 2008 (thousands persons) (numeric)

`birth_2008` Number of birth in 2008 (live birth) (numeric)

`death_2008` Number of death in 2008 (death) (numeric)

`gdppps1999` Gross domestic product (Purchasing Power Standards) in 1999 (million euros) (numeric)

`gdppps2008` Gross domestic product (Purchasing Power Standards) in 2008 (million euros) (numeric)

`pop1999` Total population in 1999 (inhabitants) (numeric)

`pop2008` Total population in 2008 (inhabitants) (numeric)

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=151

`nuts2.spdf` *Nuts2 Regions*

Description

Delineations of EU administrative units (level 2, 2006 version).

Format

SpatialPolygonsDataFrame.

Details

This SpatialPolygonsDataFrame can be used with the nuts2.df data frame

Fields

`id` Unique nuts id (character)

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

nuts3.df *Nuts3 Dataset*

Description

This dataset contains some socio-economic data

Details

This data frame can be used with the SpatialPolygonsDataFrame nuts3.spdf

Fields

`id` Unique nuts id (character)

`birth_2008` Number of birth in 2008 (live birth) (numeric)

`death_2008` Number of death in 2008 (death) (numeric)

`gdppps1999` Gross domestic product (Purchasing Power Standards) in 1999 (million euros) (numeric)

`gdppps2008` Gross domestic product (Purchasing Power Standards) in 2008 (million euros) (numeric)

`pop1999` Total population in 1999 (inhabitants) (numeric)

`pop2008` Total population in 2008 (inhabitants) (numeric)

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=151

nuts3.spdf *Nuts3 Regions*

Description

Delineations of EU administrative units (level 3, 2006 version).

Format

SpatialPolygonsDataFrame.

Details

This SpatialPolygonsDataFrame can be used with the nuts3.df data frame

Fields

`id` Unique nuts id (character)

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

`propLinkLayer` *Proportional Links Layer*

Description

Plot a layer of proportional links. Links widths are directly proportional to values of a variable.

Usage

```
propLinkLayer(spdf, df, spdfid = NULL, spdfids, spdfide, dfid = NULL, dfids,
  dfide, var, maxlwd = 40, col, legend.pos = "bottomleft",
  legend.title.txt = var, legend.title.cex = 0.8, legend.values.cex = 0.6,
  legend.values.rnd = 0, legend.frame = FALSE, add = TRUE)
```

Arguments

`spdf`	a SpatialLinesDataFrame; a link layer.
`df`	a data frame with identifiers and a variable.
`spdfid`	unique identifier in spdf (spdfids, spdfide, dfids and dfide are not used).
`spdfids`	identifier of starting points in spdf (spdfid and dfid are not used).
`spdfide`	identifier of ending points in spdf (spdfid and dfid are not used).
`dfid`	unique identifier in df (spdfids, spdfide, dfids and dfide are not used).
`dfids`	identifier of starting points in df (spdfid and dfid are not used).
`dfide`	identifier of ending points in df (spdfid and dfid are not used).
`var`	name of the variable used to plot the links widths.
`maxlwd`	maximum size of the links.
`col`	color of the links.
`legend.pos`	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.
`legend.title.txt`	title of the legend.
`legend.title.cex`	size of the legend title.
`legend.values.cex`	size of the values in the legend.
`legend.values.rnd`	number of decimal places of the values displayed in the legend.
`legend.frame`	whether to add a frame to the legend (TRUE) or not (FALSE).
`add`	whether to add the layer to an existing plot (TRUE) or not (FALSE).

Note

Unlike most of cartography functions, identifiers fields are mandatory.

See Also

gradLinkLayer, getLinkLayer, legendPropLines

Examples

```
data("nuts2006")
# Create a link layer of the twin cities agreements
twincities.spdf <- getLinkLayer(spdf = nuts2.spdf, df = twincities[,1:2])
# Plot the links - Twin cities agreements between regions
plot(nuts0.spdf, col = "grey60",border = "grey20")
propLinkLayer(spdf = twincities.spdf, df = twincities[twincities$fij>=5,],maxlwd = 10,
              spdfids = "i", spdfide = "j",
              dfids = "i", dfide = "j",legend.pos = "topright",
              var = "fij",
              col = "#92000090", add = TRUE)
```

---

propSymbolsChoroLayer *Proportional and Choropleth Symbols Layer*

Description

Plot a proportional symbols layer with color based on a quantitative data discretization.

Usage

```
propSymbolsChoroLayer(spdf, df, spdfid = NULL, dfid = NULL, var,
  inches = 0.3, fixmax = NULL, symbols = "circle", border = "grey20",
  lwd = 1, var2, breaks = NULL, method = "quantile", nclass = NULL,
  col = NULL, colNA = "white", legend.title.cex = 0.8,
  legend.values.cex = 0.6, legend.var.pos = "right",
  legend.var.title.txt = var, legend.var.values.rnd = 0,
  legend.var.style = "c", legend.var.frame = FALSE,
  legend.var2.pos = "topright", legend.var2.title.txt = var2,
  legend.var2.values.rnd = 2, legend.var2.nodata = "no data",
  legend.var2.frame = FALSE, add = TRUE, k = NULL)
```

Arguments

spdf	SpatialPointsDataFrame or SpatialPolygonsDataFrame; if spdf is a SpatialPolygonsDataFrame symbols are plotted on centroids.
df	a data frame that contains the values to plot. If df is missing spdf@data is used instead.
spdfid	identifier field in spdf, default to the first column of the spdf data frame. (optional)

`dfid` identifier field in df, default to the first column of df. (optional)

`var` name of the numeric field in df to plot the symbols sizes.

`inches` size of the biggest symbol (radius for circles, width for squares, height for bars) in inches.

`fixmax` value of the biggest symbol (see propSymbolsLayer Details).

`symbols` type of symbols, one of "circle", "square" or "bar".

`border` color of symbols borders.

`lwd` width of symbols borders.

`var2` name of the numeric field in df to plot the colors.

`breaks` break points in sorted order to indicate the intervals for assigning the colors. Note that if there are nlevel colors (classes) there should be (nlevel+1) breakpoints (see choroLayer Details).

`method` a discretization method; one of "sd", "equal", "quantile", "fisher-jenks", "q6" or "geom" (see choroLayer Details).

`nclass` a targeted number of classes. If null, the number of class is automatically defined (see choroLayer Details).

`col` a vector of colors. Note that if breaks is specified there must be one less colors specified than the number of break.

`colNA` no data color.

`legend.title.cex`
size of the legend title.

`legend.values.cex`
size of the values in the legend.

`legend.var.pos` position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.var.pos is "n" then the legend is not plotted.

`legend.var.title.txt`
title of the legend (proportional symbols).

`legend.var.values.rnd`
number of decimal places of the values in the legend.

`legend.var.style`
either "c" or "e". The legend has two display styles.

`legend.var.frame`
whether to add a frame to the legend (TRUE) or not (FALSE).

`legend.var2.pos`
position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.var2.pos is "n" then the legend is not plotted.

`legend.var2.title.txt`
title of the legend (colors).

`legend.var2.values.rnd`
number of decimal places of the values in the legend.

legend.var2.nodata
: text for "no data" values

legend.var2.frame
: whether to add a frame to the legend (TRUE) or not (FALSE).

add
: whether to add the layer to an existing plot (TRUE) or not (FALSE).

k
: share of the map occupied by the biggest symbol (this argument is deprecated; please use inches instead.).

See Also

legendBarsSymbols, legendChoro, legendCirclesSymbols, legendSquaresSymbols, choroLayer, propSymbolsLayer

Examples

```
data("nuts2006")
## Example 1
# Growth rate
nuts0.df$cagr <- (((nuts0.df$pop2008 / nuts0.df$pop1999)^(1/9)) - 1) * 100
# Countries plot
plot(nuts0.spdf, col = "grey60",border = "grey20", add=FALSE)
# Plot the symbols
propSymbolsChoroLayer(spdf = nuts0.spdf, df = nuts0.df,symbols = "circle",
                      var = "pop2008", var2 = "cagr")

## Example 2
# Growth rate at nuts2 level
nuts2.df$cagr <- (((nuts2.df$pop2008 / nuts2.df$pop1999)^(1/9)) - 1) * 100

# First layout
layoutLayer(title="Demographic trends, 1999-2008",
            scale = NULL,col = NA, coltitle = "black",
            sources = "", author = "",
            frame = FALSE, bg = "#A6CAE0",
            south = TRUE, extent = nuts0.spdf)
plot(world.spdf, col  = "#E3DEBF", border=NA, add=TRUE)
plot(nuts2.spdf, col = "grey60",border = "white", lwd=0.4, add=TRUE)

# Add some NA values
nuts2.df[1:10,"pop2008"] <- NA
nuts2.df[100:110,"cagr"] <- NA

# Plot symbols
propSymbolsChoroLayer(spdf = nuts2.spdf, df = nuts2.df,
                      var = "pop2008", var2 = "cagr",
                      inches = 0.1,
                      col = carto.pal(pal1 = "blue.pal", n1 = 2,
                                      pal2 = "red.pal", n2 = 4),
                      breaks = c(-2.43,-1,0,0.5,1,2,3.1),
                      border = "grey50", lwd = 1,
                      legend.var.pos = "topright", legend.var2.pos = "right",
                      legend.var2.title.txt = "Compound annual\ngrowth rate",
```

```
                    legend.var.title.txt = "Total Population",
                    legend.var.style = "e")
# Second layout
layoutLayer(title = "", author = "Eurostat, 2011",
            sources = "", frame ="", col = NA)
```

---

propSymbolsLayer *Proportional Symbols Layer*

Description

Plot a proportional symbols layer.

Usage

```
propSymbolsLayer(spdf, df, spdfid = NULL, dfid = NULL, var, inches = 0.3,
  fixmax = NULL, breakval = NULL, symbols = "circle", col = "#E84923",
  col2 = "#7DC437", border = "black", lwd = 1,
  legend.pos = "bottomleft", legend.title.txt = var,
  legend.title.cex = 0.8, legend.values.cex = 0.6, legend.values.rnd = 0,
  legend.style = "c", legend.frame = FALSE, add = TRUE, k = NULL)
```

Arguments

spdf	a SpatialPointsDataFrame or a SpatialPolygonsDataFrame; if spdf is a SpatialPolygonsDataFrame symbols are plotted on centroids.
df	a data frame that contains the values to plot. If df is missing spdf@data is used instead.
spdfid	identifier field in spdf, default to the first column of the spdf data frame. (optional)
dfid	identifier field in df, default to the first column of df. (optional)
var	name of the numeric field in df to plot.
inches	size of the biggest symbol (radius for circles, width for squares, height for bars) in inches.
fixmax	value of the biggest symbol (see Details).
breakval	breaking value (see Details).
symbols	type of symbols, one of "circle", "square" or "bar".
col	color of symbols.
col2	second color of symbols (see Details).
border	color of symbols borders.
lwd	width of symbols borders.
legend.pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.

`legend.title.txt`
: title of the legend.

`legend.title.cex`
: size of the legend title.

`legend.values.cex`
: size of the values in the legend.

`legend.values.rnd`
: number of decimal places of the values displayed in the legend.

`legend.style`
: either "c" or "e". The legend has two display styles, "c" stands for compact and "e" for extended.

`legend.frame`
: boolean; whether to add a frame to the legend (TRUE) or not (FALSE).

`add`
: whether to add the layer to an existing plot (TRUE) or not (FALSE).

`k`
: share of the map occupied by the biggest symbol (this argument is deprecated; please use inches instead.).

Details

The breakval parameter allows to plot symbols of two colors: the first color (col) for values superior or equal to breakval, second color (col2) for values inferior to breakval.

Two maps with the same inches and fixmax parameters will be comparable.

See Also

legendBarsSymbols, legendCirclesSymbols, legendSquaresSymbols, propSymbolsChoroLayer, propSymbolsTypoLayer

Examples

```
data("nuts2006")
## Example 1
# Layout plot
layoutLayer(title = "Countries Population in Europe",
            sources = "Eurostat, 2008",
            scale = NULL,
            frame = TRUE,
            col = "black",
            coltitle = "white",
            bg = "#D9F5FF",
            south = TRUE,
            extent = nuts0.spdf)
# Countries plot
plot(nuts0.spdf, col = "grey60",border = "grey20", add=TRUE)
# Population plot on proportional symbols
propSymbolsLayer(spdf = nuts0.spdf, df = nuts0.df,
                 var = "pop2008",
                 symbols = "square", col =  "#920000",
                 legend.pos = "right",
                 legend.title.txt = "Total\npopulation (2008)",
                 legend.style = "c")
```

```
## Example 2
# Countries plot
plot(nuts0.spdf, col = "grey60",border = "grey20")
# Population plot on proportional symbols
propSymbolsLayer(spdf = nuts0.spdf, df = nuts0.df,
                 var = "gdppps2008",
                 symbols = "bar", col =  "#B00EF0",
                 legend.pos = "right",
                 legend.title.txt = "GDP\nin Millions PPS (2008)",
                 legend.style = "e")

## Example 3
oldpar <- par(mfrow = c(1,2), mar = c(0,0,0,0))
# Countries plot
plot(nuts0.spdf, col = "grey60",border = "grey20", add=FALSE)
# Population plot on proportional symbols
propSymbolsLayer(spdf = nuts0.spdf, df = nuts0.df,
                 var = "birth_2008",
                 fixmax = max(nuts0.df$birth_2008),
                 inches = 0.2,
                 symbols = "circle", col =  "orange",
                 legend.pos = "right",
                 legend.title.txt = "nb of births",
                 legend.style = "e")
plot(nuts0.spdf, col = "grey60",border = "grey20", add=FALSE)
# Population plot on proportional symbols
propSymbolsLayer(spdf = nuts0.spdf, df = nuts0.df,
                 var = "death_2008",
                 symbols = "circle", col =  "pink",
                 fixmax = max(nuts0.df$birth_2008),
                 inches = 0.2,
                 legend.pos = "right",
                 legend.style = "e",
                 legend.title.txt = "nb of deaths")
par(oldpar)

## Example 4
nuts0.df$balance <- nuts0.df$birth_2008-nuts0.df$death_2008
plot(nuts0.spdf, col = "grey60",border = "grey20", add=FALSE)
# Population plot on proportional symbols
propSymbolsLayer(spdf = nuts0.spdf, df = nuts0.df, inches = 0.3,
                 var = "balance",
                 symbols = "circle",
                 col = "orange", col2 = "green", breakval=0,
                 legend.pos = "right",
                 legend.style = "c",
                 legend.title.txt = "Natural Balance\n(2008)")
```

---

propSymbolsTypoLayer *Proportional Symbols Typo Layer*

Description

Plot a proportional symbols layer with colors based on qualitative data.

Usage

```
propSymbolsTypoLayer(spdf, df, spdfid = NULL, dfid = NULL, var,
  inches = 0.3, fixmax = NULL, symbols = "circle", border = "grey20",
  lwd = 1, var2, col = NULL, colNA = "white", legend.title.cex = 0.8,
  legend.values.cex = 0.6, legend.var.pos = "bottomleft",
  legend.var.title.txt = var, legend.values.rnd = 0,
  legend.var.style = "c", legend.var.frame = FALSE,
  legend.var2.pos = "topright", legend.var2.title.txt = var2,
  legend.var2.values.order = NULL, legend.var2.nodata = "no data",
  legend.var2.frame = FALSE, add = TRUE, k = NULL)
```

Arguments

spdf	SpatialPointsDataFrame or SpatialPolygonsDataFrame; if spdf is a SpatialPolygonsDataFrame symbols are plotted on centroids.
df	a data frame that contains the values to plot. If df is missing spdf@data is used instead.
spdfid	identifier field in spdf, default to the first column of the spdf data frame. (optional)
dfid	identifier field in df, default to the first column of df. (optional)
var	name of the numeric field in df to plot the symbols sizes.
inches	size of the biggest symbol (radius for circles, width for squares, height for bars) in inches.
fixmax	value of the biggest symbol. (optional)
symbols	type of symbols, one of "circle", "square" or "bar".
border	color of symbols borders.
lwd	width of symbols borders.
var2	name of the factor (or character) field in df to plot.
col	a vector of colors.
colNA	no data color.
legend.title.cex	size of the legend title.
legend.values.cex	size of the values in the legend.
legend.var.pos	position of the legend for var, one of "topleft", "top",
legend.var.title.txt	title of the legend (numeric data).
legend.values.rnd	number of decimal places of the values in the legend.

legend.var.style
: either "c" or "e". The legend has two display styles, "c" stands for compact and "e" for extended.

legend.var.frame
: whether to add a frame to the legend (TRUE) or not (FALSE).

legend.var2.pos
: position of the legend for var2, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright".

legend.var2.title.txt
: title of the legend (factor data).

legend.var2.values.order
: values order in the legend, a character vector that matches var modalities. Colors will be affected following this order.

legend.var2.nodata
: text for "no data" values

legend.var2.frame
: whether to add a frame to the legend (TRUE) or not (FALSE).

add
: whether to add the layer to an existing plot (TRUE) or not (FALSE).

k
: share of the map occupied by the biggest symbol (this argument is deprecated; please use inches instead.).

See Also

legendBarsSymbols, legendTypo, legendCirclesSymbols, legendSquaresSymbols, typoLayer, propSymbolsLayer

Examples

```
data("nuts2006")
## Example 1
plot(nuts0.spdf, col = "grey60",border = "grey20")
nuts0.df$typo <- c(rep("A",10),rep("B",10),rep("C",10),rep("D",4))
propSymbolsTypoLayer(spdf = nuts0.spdf, df = nuts0.df,
                     var = "pop2008", var2="typo")

## Example 2
#Countries plot
plot(nuts0.spdf, col = "grey60",border = "grey20", add = FALSE)
nuts0.df$typo <- c(rep("A",10),rep("B",10),rep("C",10),rep("D",4))
nuts0.df$typo[1:3] <- NA
nuts0.df$pop2008[4:6] <- NA
propSymbolsTypoLayer(spdf = nuts0.spdf, df = nuts0.df,
                     var = "pop2008", var2="typo",
                     symbols = "circle",
                     legend.var.pos = "topright",
                     legend.var2.pos = "right",
                     legend.var.title.txt = "Total\npopulation (2008)",
                     legend.values.rnd = -3,
                     legend.var2.title.txt = "Category",
```

```
                    col = carto.pal(pal1 = "pastel.pal", 4),
                    legend.var2.values.order = c("D", "A", "B", "C"),
                    legend.var.style = "c")
# Layout plot
layoutLayer(title = "Countries Population & Color in Europe",
            sources = "UMS RIATE, 2015",
            scale = NULL,
            frame = TRUE,
            col = "black",
            coltitle = "white")
```

---

propTrianglesLayer *Double Proportional Triangle Layer*

Description

Plot a double proportional triangles layer.

Usage

```
propTrianglesLayer(spdf, df, spdfid = NULL, dfid = NULL, var1,
  col1 = "#E84923", var2, col2 = "#7DC437", k = 0.02,
  legend.pos = "topright", legend.title.txt = paste(var1, var2, sep =
  " / "), legend.title.cex = 0.8, legend.var1.txt = var1,
  legend.var2.txt = var2, legend.values.cex = 0.6, legend.values.rnd = 0,
  legend.style = "c", legend.frame = FALSE, add = TRUE)
```

Arguments

spdf	a SpatialPointsDataFrame or a SpatialPolygonsDataFrame; if spdf is a SpatialPolygonsDataFrame symbols are plotted on centroids.
df	a data frame that contains the values to plot. If df is missing spdf@data is used instead.
spdfid	identifier field in spdf, default to the first column of the spdf data frame. (optional)
dfid	identifier field in df, default to the first column of df. (optional)
var1	name of the first numeric field in df to plot, positive values only (top triangle).
col1	color of top triangles.
var2	name of the second numeric field in df to plot, positive values only (bottom triangle).
col2	color of bottom triangles.
k	share of the map occupied by the biggest symbol.
legend.pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.

`legend.title.txt`
: title of the legend.

`legend.title.cex`
: size of the legend title.

`legend.var1.txt`
: label of the top variable.

`legend.var2.txt`
: label of the bottom variable.

`legend.values.cex`
: size of the values in the legend.

`legend.values.rnd`
: number of decimal places of the values displayed in the legend.

`legend.style`
: either "c" or "e". The legend has two display styles, "c" stands for compact and "e" for extended.

`legend.frame`
: boolean; whether to add a frame to the legend (TRUE) or not (FALSE).

`add`
: whether to add the layer to an existing plot (TRUE) or not (FALSE).

See Also

legendPropTriangles

Examples

```
data("nuts2006")
# Example 1
plot(nuts0.spdf)
# There is no data for deaths in Turkey
propTrianglesLayer(spdf = nuts0.spdf, df = nuts0.df,
                   var1 = "birth_2008",
                   var2 = "death_2008")

# Example 2
layoutLayer(title = "Births and Deaths in Europe, 2008",
            sources = "",author = "",
            scale = NULL,
            frame = FALSE,
            col = "black",
            coltitle = "white",
            extent = nuts0.spdf)
plot(countries.spdf,col="#E0E0E0",border="white",lwd=1, add=TRUE)
plot(nuts0.spdf,col="#E5CFC1",border="white",lwd=2,add=TRUE)
# There is no data for deaths in Turkey
propTrianglesLayer(spdf = nuts0.spdf, df = nuts0.df,
                   var1 = "birth_2008", legend.style = "e",
                   var2 = "death_2008", legend.frame = TRUE,
                   col1="#FF9100",col2="#45C945",k = 0.1, add=TRUE)
```

smoothLayer *Smooth Layer*

Description

Plot a layer of smoothed data. It can also compute a ratio of potentials.

This function is a wrapper around the quickStewart function in SpatialPosition package.

The SpatialPosition package also provides:

- vignettes to explain the computation of potentials;
- more customizable inputs and outputs (custom distance matrix, raster output...);
- other functions related to spatial interactions (Reilly and Huff catchment areas).

Usage

```
smoothLayer(spdf, df, spdfid = NULL, dfid = NULL, var, var2 = NULL,
  typefct = "exponential", span, beta, resolution = NULL, mask = NULL,
  nclass = 8, breaks = NULL, col = NULL, border = "grey20", lwd = 1,
  legend.pos = "bottomleft", legend.title.txt = "Potential",
  legend.title.cex = 0.8, legend.values.cex = 0.6, legend.values.rnd = 0,
  legend.frame = FALSE, add = FALSE)
```

Arguments

spdf	a SpatialPolygonsDataFrame.
df	a data frame that contains the values to compute If df is missing spdf@data is used instead.
spdfid	name of the identifier field in spdf, default to the first column of the spdf data frame. (optional)
dfid	name of the identifier field in df, default to the first column of df. (optional)
var	name of the numeric field in df used to compute potentials.
var2	name of the numeric field in df used to compute potentials. This field is used for ratio computation (see Details).
typefct	character; spatial interaction function. Options are "pareto" (means power law) or "exponential". If "pareto" the interaction is defined as: (1 + alpha * mDistance) ^ (-beta). If "exponential" the interaction is defined as: exp(- alpha * mDistance ^ beta). The alpha parameter is computed from parameters given by the user (beta and span).
span	numeric; distance where the density of probability of the spatial interaction function equals 0.5.
beta	numeric; impedance factor for the spatial interaction function.
resolution	numeric; resolution of the output SpatialPointsDataFrame (in map units).

mask	SpatialPolygonsDataFrame; mask used to clip contours of potentials.
nclass	numeric; a targeted number of classes (default to 8). Not used if breaks is set.
breaks	numeric; a vector of values used to discretize the potentials.
col	a vector of colors. Note that if breaks is specified there must be one less colors specified than the number of break.
border	color of the polygons borders.
lwd	borders width.
legend.pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.
legend.title.txt	title of the legend.
legend.title.cex	size of the legend title.
legend.values.cex	size of the values in the legend.
legend.values.rnd	number of decimal places of the values in the legend.
legend.frame	whether to add a frame to the legend (TRUE) or not (FALSE).
add	whether to add the layer to an existing plot (TRUE) or not (FALSE).

Details

If var2 is provided the ratio between the potentials of var (numerator) and var2 (denominator) is computed.

Value

An `invisible` SpatialPolygonsDataFrame is returned (see `quickStewart`).

See Also

quickStewart, SpatialPosition, choroLayer

Examples

```
## Not run:
data("nuts2006")

# Potential of GDP
smoothLayer(spdf = nuts3.spdf, df = nuts3.df,
            var = 'gdppps2008',
            span = 75000, beta = 2,
            mask = nuts0.spdf,
            legend.title.txt = "GDP",
            legend.pos = "topright", legend.values.rnd = -2)
```

```
# Potential of GDP per Capita
nuts3.df$gdppps2008 <- nuts3.df$gdppps2008 * 1000000
smoothLayer(spdf = nuts3.spdf, df = nuts3.df,
            var = 'gdppps2008', var2 = 'pop2008',
            span = 75000, beta = 2,
            mask = nuts0.spdf,
            legend.title.txt = "GDP PER CAPITA",
            legend.pos = "topright", legend.values.rnd = -2)

## End(Not run)
```

`tilesLayer` *Plot Tiles from Open Map Servers*

Description

Plot tiles from open map servers.

Usage

```
tilesLayer(x, add = FALSE)
```

Arguments

x	a RasterBrick object; the getTiles function outputs these objects.
add	whether to add the layer to an existing plot (TRUE) or not (FALSE).

Note

This function is a wrapper for plotRGB from the raster package.

See Also

getTiles

Examples

```
## Not run:
data("nuts2006")
# extract Denmark
spdf <- nuts0.spdf[nuts0.spdf$id=="DK",]
# Download the tiles, extent = Denmark
den <- getTiles(spdf = spdf, type = "osm", crop = TRUE)
class(den)
# Plot the tiles
tilesLayer(den)
# Plot countries
plot(spdf, add=TRUE)
# Map tiles sources
```

```
mtext(text = "Map data © OpenStreetMap contributors, under CC BY SA.",
      side = 1, adj = 0, cex = 0.7, font = 3)

## End(Not run)
```

`twincities` *Twin Cities Dataset*

Description

This dataset contains the number of international twinning agreements betwwen cities. Agreements are aggregated at nuts2 level.

Details

This data frame can be used with the SpatialPolygonsDataFrame nuts2.spdf

Fields

`i` nuts2 identifier

`j` nuts2 identifier

`fij` number of agreements

Source

Adam Ploszaj - Centre for European Regional and Local Studies EUROREG, University of Warsaw, Poland. Primary source: Wikipedia, 2011.

`typoLayer` *Typology Layer*

Description

Plot a typology layer.

Usage

```
typoLayer(spdf, df, spdfid = NULL, dfid = NULL, var, col = NULL,
  border = "grey20", lwd = 1, colNA = "white",
  legend.pos = "bottomleft", legend.title.txt = var,
  legend.title.cex = 0.8, legend.values.cex = 0.6,
  legend.values.order = NULL, legend.nodata = "no data",
  legend.frame = FALSE, add = FALSE)
```

Arguments

spdf	a SpatialPolygonsDataFrame.
df	a data frame that contains the values to plot. If df is missing spdf@data is used instead.
spdfid	identifier field in spdf, default to the first column of the spdf data frame. (optional)
dfid	identifier field in df, default to the first column of df. (optional)
var	name of the field in df to plot.
col	a vector of colors.
border	color of the polygons borders.
lwd	borders width.
colNA	no data color.
legend.pos	position of the legend, one of "topleft", "top", "topright", "left", "right", "bottomleft", "bottom", "bottomright". If legend.pos is "n" then the legend is not plotted.
legend.title.txt	title of the legend.
legend.title.cex	size of the legend title.
legend.values.cex	size of the values in the legend.
legend.values.order	values order in the legend, a character vector that matches var modalities. Colors will be affected following this order.
legend.nodata	no data label.
legend.frame	whether to add a frame to the legend (TRUE) or not (FALSE).
add	whether to add the layer to an existing plot (TRUE) or not (FALSE).

See Also

propSymbolsTypoLayer, typoLayer, legendTypo

Examples

```
data(nuts2006)
## Example 1
nuts0.df$typo <- c(rep("A",10),rep("B",10),rep("C",10),rep("D",4))
typoLayer(spdf = nuts0.spdf, df = nuts0.df, var = "typo")


## Example 2
nuts0.df$typo <- c(rep("A",10),rep("B",10),rep("C",10),rep("D",4))
typoLayer(spdf = nuts0.spdf, df = nuts0.df,
          var="typo",  col = carto.pal(pal1 = "multi.pal", 4),
          legend.values.order = c("D", "B", "A", "C"),
```

```
          legend.pos = "topright",
          legend.title.txt = "Category")
layoutLayer(title = "Colors in Europe",
            sources = "UMS RIATE, 2015",
            scale = NULL,
            frame = TRUE,
            col = "black",
            coltitle = "white")
```

`world.spdf` *World Background*

Description

World background.

Format

SpatialPolygonsDataFrame.

Source

UMS RIATE - http://www.ums-riate.fr/Webriate/?page_id=153

Index