Package ‘cartography’

February 7, 2019

Title Thematic Cartography

Version 2.2.0

Description Create and integrate maps in your R workflow. This package helps to design cartographic representations such as proportional symbols, choropleth, typology, flows or discontinuities maps. It also offers several features that improve the graphic presentation of maps, for instance, map palettes, layout elements (scale, north arrow, title...), labels or legends. See Giraud and Lambert (2017) <doi:10.1007/978-3-319-57336-6_13>.

License GPL-3

URL https://github.com/riatelab/cartography/

BugReports https://github.com/riatelab/cartography/issues/

LazyData true

Depends R (>= 3.3.0)

Imports classInt, stats, graphics, methods, rosm, raster, Rcpp, rgeos, sp (>= 1.2-4), sf (>= 0.5-4)

Suggests SpatialPosition, knitr, rmarkdown, rgdal, testthat

LinkingTo Rcpp

VignetteBuilder knitr

SystemRequirements GDAL (>= 2.0.1), GEOS (>= 3.4.0), PROJ.4 (>= 4.8.0)

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation yes

Author Timothée Giraud [cre, aut],
Nicolas Lambert [aut],
Ian Fellows [cph] (no overlap algorithm for labels, from wordcloud package)

Maintainer Timothée Giraud <timothee.giraud@cnrs.fr>

Repository CRAN

Date/Publication 2019-02-07 12:13:29 UTC
R topics documented:

barscale .................................................. 3
carto.pal .................................................. 4
carto.pal.info ............................................ 5
cartography ................................................ 6
choroLayer ................................................ 7
coasts.spdf .............................................. 9
countries.spdf .......................................... 9
discLayer .................................................. 10
display.carto.all ........................................ 12
display.carto.pal ........................................ 12
dotDensityLayer .......................................... 13
frame.spdf ................................................ 15
getBorders ................................................ 15
getBreaks .................................................. 16
getFigDim .................................................. 18
getGridData ............................................... 19
getGridLayer .............................................. 19
getLinkLayer ............................................. 20
getOuterBorders ......................................... 21
getPencilLayer .......................................... 22
getTiles ................................................... 23
gradLinkLayer ............................................ 25
gradLinkTypoLayer ....................................... 26
graticule.spdf ........................................... 28
labelLayer ............................................... 29
layoutLayer .............................................. 30
legendBarsSymbols ...................................... 31
legendChoro ............................................. 32
legendCirclesSymbols ................................... 34
legendGradLines ......................................... 35
legendPropLines ......................................... 36
legendPropTriangles ..................................... 37
legendSquaresSymbols ................................... 38
legendTypo ............................................... 39
north ...................................................... 40
nuts0.df .................................................. 41
nuts0.spdf ............................................... 41
nuts1.df .................................................. 42
nuts1.spdf ............................................... 43
nuts2.df .................................................. 43
nuts2.spdf ............................................... 44
nuts3.df .................................................. 45
nuts3.spdf ............................................... 45
propLinkLayer ........................................... 46
propSymbolsChoroLayer ................................ 47
propSymbolsLayer ........................................ 50
barscale

Description
Plot a scale bar.

Usage
barscale(size, lwd = 1.5, cex = 0.6, pos = "bottomright", style = "pretty")

Arguments
- **size**: size of the scale bar in kilometers. If size is not set, an automatic size is used (1/10 of the map width).
- **lwd**: width of the scale bar.
- **cex**: cex of the text.
- **pos**: position of the legend, default to "bottomright". "bottomright" or a vector of two coordinates (c(x, y)) are possible.
- **style**: style of the legend, either "pretty" or "oldschool". The "oldschool" style only uses the "size" parameter.

Note
This scale bar is not accurate on unprojected (long/lat) maps.

See Also
layoutLayer

Examples
library(sf)
mtq <- st_read(system.file("pkg/mrq.qpkg", package="cartography"))
plot(st_geometry(mtq), col = "grey60", border = "grey20")
barscale(size = 5)
barscale(size = 5, lwd = 2, cex = .9, pos = c(14000, 159600))
Build Cartographic Palettes

Description

Build sequential, diverging and qualitative color palettes. Diverging color palettes can be dissymmetric (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 gradient (see Details).
- `n1`: number of colors (up to 20).
- `pal2`: name of the color gradient (see Details).
- `n2`: number of colors (up to 20).
- `middle`: a logical value. If TRUE, a neutral color ("#F6F6F6", light grey) between two gradients is added.
- `transparency`: a logical value. If TRUE, contrasts are enhanced 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](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

```r
# 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 asymmetric
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")
```

Display the Names of all Cartographic Palettes

Description

Display the names of all color palettes.
Usage
carto.pal.info()

Value
A vector of color palettes names is returned.

See Also
carto.pal, display.carto.pal, display.carto.all

Examples
carto.pal.info()

cartography Cartography Package

Description
This package helps to design cartographic representations such as proportional symbols, choropleth,
typology, flows or discontinuities maps. It also offers several features that improve the graphic
presentation of maps, for instance, map palettes, layout elements (scale, north arrow, title...), labels
or legends.

A vignette contains commented scripts on how to create various maps and a cheat sheet displays a
quick overview of cartography’s main features:
- vignette(topic = "cartography", package = "cartography");
- vignette(topic = "cheatsheet", package = "cartography").

Main functions:
• Proportional symbols maps (circles, squares, bars)
  propSymbolsLayer, propSymbolsChoroLayer, propSymbolsTypoLayer, propTrianglesLayer
• Choropleth maps (main classification methods are available)
  choroLayer
• Typology maps
  typoLayer
• Flow maps (proportional and classified links)
  getLinkLayer, propLinkLayer, gradLinkLayer, gradLinkTypoLayer
• Discontinuities maps
  getBorders, discLayer
• Cartographic palettes
  carto_pal
• Layout (scale, north arrow, title...)
  layoutLayer, north, barscale
• Labels
  labelLayer

• Legends
  legendBarsSymbols, legendChoro, legendCirclesSymbols, legendGradLines, legendPropLines,
  legendPropTriangles, legendSquaresSymbols, legendTypo

• Access to cartographic APIs (via rosm package)
  getTiles, tilesLayer

• Irregular polygons to regular grid, transformation with data handling
  getGridLayer

---

**choroLayer**

**Choropleth Layer**

**Description**

Plot a choropleth layer.

**Usage**

```r
choroLayer(x, 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, legend.border = "black", 
  legend.horiz = FALSE, add = FALSE)
```

**Arguments**

- `x` an sf object, a simple feature collection. If x is used then spdf, df, spdfid and dfid are not.
- `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 x or 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 classification method; one of "sd", "equal", "quantile", "fisher-jenks", "q6", "geom", "arith", "em" or "msd" (see `getBreaks`).
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", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)). 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).

legend.border color of boxes borders in the legend.

legend.horiz whether to display the legend horizontally (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.

If breaks is defined as c(2, 5, 10, 15, 20) intervals will be: [2 - 5], [5 - 10], [10 - 15], [15 - 20].

References


See Also

getBreaks, carto_pal, legendChoro, propSymbolsChoroLayer
Examples

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
# Population density
mtq$POPdens <- 1e6 * mtq$POP / st_area(x = mtq)

# Default
choroLayer(x = mtq, var = "POPdens")

# With parameters
choroLayer(x = mtq, var = "POPdens",
           method = "quantile", nclass = 5,
           col = carto_pal(pal1 = "sand.pal", n1 = 5),
           border = "grey40",
           legend.pos = "topright", legend.values.rnd = 0,
           legend.title.txt = "Population Density\n(people per km2)"
)

# Layout
layoutLayer(title = "Population Distribution in Martinique, 2015")
```

---

coasts.spdf  

**Coastline of Europe**

Description

Coastline of Europe.

Format

SpatialLinesDataFrame.

Source

UMS RIATE - [http://riate.cnrs.fr/?page_id=153](http://riate.cnrs.fr/?page_id=153)

countries.spdf  

**Countries in the European Area**

Description

Countries in the European area.

Format

SpatialPolygonsDataFrame.

Source

UMS RIATE - [http://riate.cnrs.fr/?page_id=153](http://riate.cnrs.fr/?page_id=153)
**discLayer**  
*Discontinuities Layer*

**Description**

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

**Usage**

```r
discLayer(x, df, 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, spdf, spdfid1, spdfid2)
```

**Arguments**

- `x`: an sf object, a simple feature collection, as outputted by the `getBorders` function.
- `df`: a data frame that contains the values used to compute and plot discontinuities.
- `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 classification method; one of "sd", "equal", "quantile", "fisher-jenks"," q6", "geom", "arith", "em" or "msd" (see `getBreaks`).
- `nclass`: a targeted number of classes. If null, the number of class is automatically defined (see `getBreaks`).
- `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", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)). 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.
discLayer

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).
spdf
   defunct.
spdfid1
   defunct.
spdfid2
   defunct.

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 sf object (MULTISTRING) with the discontinuity measures is returned.

See Also

getBorders, gradLinkLayer, legendGradLines

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
# Get borders
mtq.borders <- getBorders(x = mtq)
# Median Income
choroLayer(x = mtq, var = "MED", border = "grey", lwd = 0.5,
   method = 'equal', nclass = 6, legend.pos = "topleft",
   legend.title.txt = "Median Income
(in euros)"
)
# Discontinuities
discLayer(x = mtq.borders, df = mtq,
   var = "MED", col="red4", nclass=3,
   method="equal", threshold = 0.4, sizemin = 0.5,
   sizemax = 10, type = "abs", legend.values.rnd = 0,
   legend.title.txt = "Discontinuities
(absolute difference)",
   legend.pos = "bottomleft", add=TRUE)
display.carto.all  

**Display all Cartographic Palettes**  

**Description**  
Display all the available color palettes.  

**Usage**  
```r  
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**  
```r  
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**  
```r  
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")

---

**Description**

Plot a dot density layer.

**Usage**

dotDensityLayer(x, 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**

- **x**: an sf object, a simple feature collection. If x is used then spdf, df, spdfid and dfid are not.
- **spdf**: a SpatialPolygonsDataFrame.
- **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**

```r
library(sf)
mtq <- st_read(system.file("pkg/mtq.pkg", package="cartography"))
plot(st_geometry(mtq), col = "#B8704D50")
dotDensityLayer(x = mtq, var="POP", pch=20, col = "#ed4", n = 50)
layoutLayer(title = "Population Distribution in Martinique, 2015")
```
frame.spdf  

Frame around Europe

Description
Frame around European countries.

Format
SpatialPolygonsDataFrame.

Source
UMS RIATE - http://riate.cnrs.fr/?page_id=153

getBorders  

Extract Polygons Borders

Description
Extract borders between polygons.

Usage
getBorders(x, id, spdf, spdfid = NULL)

Arguments

x  an sf object, a simple feature collection or a SpatialPolygonsDataFrame.

id  identifier field in x or spdf, default to the first column. (optional)

spdf  deprecated, a SpatialPolygonsDataFrame. This SpatialPolygonsDataFrame has to be projected (planar coordinates).

spdfid  deprecated, identifier field in spdf, default to the first column of the spdf data frame. (optional)

Value
An sf object (MULTILINESTRING) 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

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
# Get borders
mtq.borders <- getBorders(x = mtq)
# Plot polygons
plot(st_geometry(mtq), border = NA, col = "grey60")
# Plot borders
plot(st_geometry(mtq.borders),
     col = sample(x = rainbow(nrow(mtq.borders))),
     lwd = 3, add = TRUE)
```

getBreaks

### Classification

**Description**

A function to classify continuous variables.

**Usage**

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

**Arguments**

- `v`: a vector of numeric values.
- `nclass`: a number of classes.
- `method`: a classification method; one of "sd", "equal", "quantile", "fisher-jenks","q6", "geom", "arith", "em" 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 "arith" method is based on an arithmetic progression along the variable values.

The "em" method is based on nested averages computation.

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 classInt::classIntervals + arith, em, q6, geom and msd methods.

**Examples**

```r
library(sf)
mtq <- st_read(system.file("pkg/mqpkg", package="cartography"))
var <- mtq$MED
# Histogram
hist(var, probability = TRUE, breaks = 20)
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 = "#0D9F9")
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 = "#0D9F9")
rug(var)

# Mean and standard deviation (msd)
breaks <- getBreaks(v = var, method = "msd", k = 1, middle = TRUE)
hist(var, probability = TRUE, breaks = breaks, col = "#0D9F9")
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 or vector format.
Output dimension are based on a spatial object dimension ratio, margins of the figure, a targeted
width or height and a resolution.

Usage

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

Arguments

x an sf object, a simple feature collection or a Spatial*DataFrame.
spdf deprecated, a Spatial*DataFrame.
width width of the figure (in pixels), either width or height must be set.
height height 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

```R
## Not run:
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))

## PNG export
# get figure dimension
sizes <- getFigDim(x = mtq, width = 450, mar = c(0,0,1.2,0))
# export the map
png(filename = "mtq.png", width = sizes[1], height = sizes[2])
par(mar = c(0,0,1.2,0))
plot(st_geometry(mtq), col = "#D1914D", border = "white", bg = "#A6CAE0")
title("Madina")
dev.off()
```
getGridData

## Compute Data for a Grid Layer

### Description

Defunct

### Usage

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

### Arguments

- `x`: ...
- `df`: ...
- `dfid`: ...
- `var`: ...

getGridLayer

## Build a Regular Grid Layer

### Description

Build a regular grid based on an sf object or a SpatialPolygonsDataFrame.

### Usage

```r
getGridLayer(x, cellsize, type = "regular", var, spdf, spdfid = NULL)
```
getLinkLayer

Create a Links Layer from a Data Frame of Links.

Arguments

- **x**: an sf object, a simple feature collection or a SpatialPolygonsDataFrame.
- **cellsize**: targeted area of the cell, in map units.
- **type**: shape of the cell, "regular" for squares, "hexagonal" for hexagons.
- **var**: name of the numeric field(s) in x to adapt to the grid (a vector).
- **spdf**: deprecated, a SpatialPolygonsDataFrame.
- **spdfid**: deprecated, identifier field in spdf, default to the first column of the spdf data frame (optional).

Value

A grid is returned as an sf object.

Examples

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
# Plot density of population
mtq$POPdens <- 1e6 * mtq$POP / st_area(mtq)
bks <- getBreaks(v = mtq$POPdens, method = "geom", 5)
cols <- carto.pal(pal1 = "taupe.pal", n1 = 5)
opar <- par(mfrow = c(1,2), mar = c(0,0,0,0))
choroLayer(x = mtq, var = "POPdens", breaks = bks,
            border = "burlywood3", col = cols,
            legend.pos = "bottom", legend.values.rnd = 0,
            legend.title.txt = "Population density")

mygrid <- getGridLayer(x = mtq, cellsize = 3e7,
                        type = "hexagonal", var = "POP")
## conversion from square meter to square kilometers
mygrid$POPENSG <- 1e6 * mygrid$POP / mygrid$gridarea
choroLayer(x = mygrid, var = "POPENSG", breaks = bks,
            border = "burlywood3", col = cols,
            legend.pos = "n", legend.values.rnd = 1,
            legend.title.txt = "Population density")
par(opar)
```

Description

Create a links layer from a data frame of links.

Usage

```r
getLinkLayer(x, xid = NULL, df, dfid = NULL, spdf, spdf2 = NULL,
             spdfid = NULL, spdf2id = NULL, dfids = NULL, dfide = NULL)
```
getOuterBorders

Arguments

- **x**: an sf object, a simple feature collection (or a Spatial*DataFrame).
- **xid**: identifier field in x, default to the first column (optional)
- **df**: a data frame that contains identifiers of starting and ending points.
- **dfid**: identifier fields in df, character vector of length 2, default to the two first columns. (optional)
- **spdf**: defunct.
- **spdf2**: defunct.
- **spdfid**: defunct.
- **spdf2id**: defunct.
- **dfids**: defunct.
- **dfide**: defunct.

Value

An sf LINESTRING is returned, it contains two fields (origins and destinations).

See Also

gradLinkLayer, propLinkLayer

Examples

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
mob <- read.csv(system.file("csv/mob.csv", package="cartography"))
# Select links from Fort-de-France (97209)
mob_97209 <- mob[mob$i == 97209, ]
# Create a link layer
mob.sf <- getLinkLayer(x = mtq, df = mob_97209, dfid = c("i", "j"))
# Plot the links
plot(st_geometry(mtq), col = "grey")
plot(st_geometry(mob.sf), col = "red4", lwd = 2, add = TRUE)
```

getOuterBorders Extract Polygons Outer Borders

Description

Extract outer borders between polygons. Outer borders are non-contiguous polygons borders (e.g. maritime borders).

Usage

```r
getOuterBorders(x, id, res = NULL, width = NULL, spdf, spdfid = NULL)
```
getPencilLayer

Arguments

- **x**: an sf object, a simple feature collection or a SpatialPolygonsDataFrame.
- **id**: identifier field in x, default to the first column. (optional)
- **res**: resolution of the grid used to compute borders (in x units). A high resolution will give more detailed borders. (optional)
- **width**: maximum distance between used to compute borders (in x units). A higher width will build borders between units that are farther apart. (optional)
- **spdf**: deprecated, a SpatialPolygonsDataFrame. This SpatialPolygonsDataFrame has to be projected (planar coordinates).
- **spdfid**: deprecated, identifier field in spdf, default to the first column of the spdf data frame. (optional)

Value

An sf object (MULTILINESTRING) 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

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
# Get units borders
mtq.outer <- getOuterBorders(x = mtq, res = 1000, width = 2500)
# Plot municipalities
plot(st_geometry(mtq), col = "grey60")
# Plot borders
plot(st_geometry(mtq.outer), col = sample(x = rainbow(nrow(mtq.outer))),
     lwd = 3, add = TRUE)
```

getPencillayer  

**Pencil Layer**

Description

Create a pencil layer. This function transforms a POLYGON or MULTIPOLYGON sf object into a MULTILINESTRING one.
Usage

getPencilLayer(x, size = 100, buffer = 1000, lefthanded = TRUE)

Arguments

x  an sf object, a simple feature collection (POLYGON or MULTIPOLYGON).
size density of the penciling. Median number of points used to build the MULTILINESTRING.
buffer buffer around each polygon. This buffer (in map units) is used to take sample points. A negative value adds a margin between the penciling and the original polygons borders
lefthanded if TRUE the penciling is done left-handed style.

Value

A MULTILINESTRING sf object is returned.

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
mtq_pencil <- getPencilLayer(x = mtq)
plot(st_geometry(mtq_pencil), col = 1:8)
plot(st_geometry(mtq), add = TRUE)

typoLayer(x = mtq_pencil, var="STATUS",
   col = c("aquamarine4", "yellow3", "wheat"),
   legend.values.order = c("Prefecture",
                        "Sub-prefecture",
                        "Simple municipality"),
   legend.pos = "topright",
   legend.title.txt = "Status")
plot(st_geometry(mtq), add = TRUE, ldy=2)
layoutLayer(title = "Municipality Status")

getTiles  Get Tiles from Open Map Servers

Description

Get map tiles based on a spatial object extent. Maps can be fetched from various open map servers.

Usage

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

Arguments

x an sf object, a simple feature collection or a Spatial*DataFrame.

spdf deprecated, a Spatial*DataFrame with a valid projection attribute.

type the tile server from which to get the map, one of "osm", "hotstyle", "hikebike", "osmgrayscale", "stamenbw", "stamenwatercolor", "cartodark", "cartolight".

zoom the zoom level. If null, it is determined automatically (see Details).

crop TRUE if results should be cropped to the specified x extent, FALSE otherwise.

verbose if TRUE a progress bar is displayed.

Details

Zoom levels are described on the OpenStreetMap wiki: [http://wiki.openstreetmap.org/wiki/Zoom_levels](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

```r
## Not run:
library(sf)
mtq <- st_read(system.file("pkg/mtq.pkg", package="cartography"))
# Download the tiles, extent = Martinique
mtqOSM <- getTiles(x = mtq, type = "osm", crop = TRUE)
# Plot the tiles
tilesLayer(mtqOSM)
# Plot countries
plot(st_geometry(mtq), add=TRUE)
txt <- "© OpenStreetMap contributors. Tiles style under CC BY-SA, www.openstreetmap.org/copyright"
mtext(text = txt, side = 1, adj = 0, cex = 0.7, font = 3)
## End(Not run)
```
### Description

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

### Usage

```r
g.gradLinkLayer(x, df, xid = NULL, dfid = NULL, 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, spdf, spdfid, spdfids, spdfide, 
dfids, dfide)
```

### Arguments

- **x**: an sf object, a simple feature collection.
- **df**: a data frame that contains identifiers of starting and ending points and a variable.
- **xid**: identifier fields in x, character vector of length 2, default to the 2 first columns. (optional)
- **dfid**: identifier fields in df, character vector of length 2, default to the two first columns. (optional)
- **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", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)). 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).
**gradLinkTypoLayer**

**Description**

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

**Usage**

```r
ggradLinkTypoLayer(x, df, xid = NULL, dfid = NULL, 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,
```

**Examples**

```r
library(sf)
mtq <- st_read(system.file("pkg/mtg.gpkg", package="cartography"))
# Create a link layer - work mobilities to Fort-de-France (97209)
mob.sf <- getLinkLayer(x = mtq, df = mob[mob$j==97209,], dfid = c("i", "j"))
# Plot the links - Work mobility
plot(st_geometry(mtq), col = "grey60", border = "grey20")
ggradLinkTypoLayer(x = mob.sf, df = mob,
  legend.pos = "topright",
  var = "fij",
  breaks = c(109,500,1000,2000,4679),
  lwd = c(1,2,4,10),
  col = "#92000090", add = TRUE)
```

Note

Unlike most of cartography functions, identifiers fields are mandatory.

See Also

`getLinkLayer`, `propLinkLayer`, `legendGradLines`
gradLinkTypoLayer

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, spdf, spdfid, spdfids,
spdfide, dfids, dfide)

Arguments

x an sf object, a simple feature collection.
df a data frame that contains identifiers of starting and ending points and variables.
xid identifier fields in x, character vector of length 2, default to the 2 first columns.
(df) dfid identifier fields in df, character vector of length 2, default to the two first columns.
(var) 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, one of "topleft", "top", "topright", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)).
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, one of "topleft", "top", "topright", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)).
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).

spdf
defunct.

spdfid
defunct.

spdfids
defunct.

spdfide
defunct.

dfids
defunct.

dfide
defunct.

Note

Unlike most of cartography functions, identifiers fields are mandatory.

See Also

ggetLinkLayer, propLinkLayer, legendGradLines, gradLinkLayer

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
mob <- read.csv(system.file("csv/mob.csv", package="cartography"))
# Create a link layer - work mobilities to Fort-de-France (97209) and
# Le Lamentin (97213)

mob.sf <- getLinkLayer(x = mtq, df = mob[, mob$y %in% c(97209, 97213)],
   dfid = c("i", "j"))

# Plot the links - Work mobility

plot(st_geometry(mtq), col = "grey60", border = "grey20")

gradLinkTypoLayer(x = mob.sf, df = mob,
   var = "fij",
   breaks = c(109, 500, 1000, 2000, 4679),
   lwd = c(1, 2, 4, 10),
   var2='j', add = TRUE)

---

graticule.spdf

Graticule around Europe

Description

Graticule around Europe.

Format

SpatialLines.

Source

UMS RIATE - http://riate.cnrs.fr/?page_id=153
labelLayer  

**Description**

Put labels on a map.

**Usage**

```
labelLayer(x, spdf, df, spdfid = NULL, dfid = NULL, txt,
  col = "black", cex = 0.7, overlap = TRUE, show.lines = TRUE,
  halo = FALSE, bg = "white", r = 0.1, ...)
```

**Arguments**

- `x`: an sf object, a simple feature collection.
- `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.
- `overlap`: if FALSE, labels are moved so they do not overlap.
- `show.lines`: if TRUE, then lines are plotted between x,y and the word, for those words not covering their x,y coordinate
- `halo`: If TRUE, then a 'halo' is printed around the text and additional arguments bg and r can be modified to set the color and width of the halo.
- `bg`: halo color if halo is TRUE
- `r`: width of the halo
- `...`: further text arguments.

**See Also**

- `layoutLayer`
Examples

library(sf)
opar <- par(mar = c(0,0,0,0))
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st.geometry(mtq), col = "darkseagreen3", border = "darkseagreen4",
     bg = "#A6CAE0")
labellayer(x = mtq, txt = "LIBGEO", col = "black", cex = 0.7, font = 4,
         halo = TRUE, bg = "white", r = 0.1,
         overlap = FALSE, show.lines = FALSE)
par(opar)

layoutLayer  Layout Layer

Description

Plot a layout layer.

Usage

layoutLayer(title = "Title of the map, year", sources = "",
         author = "", horiz = TRUE, col = "black", coltitle = "white",
         theme = NULL, bg = NULL, scale = "auto",
         posscale = "bottomright", frame = TRUE, north = FALSE,
         south = FALSE, extent = NULL, tabtitle = FALSE,
         postitle = "left")

Arguments

title  title of the map.
sources sources of the map (or something else).
author  author of the map (or something else).
horiz  orientation of sources and author. TRUE for horizontal display on the bottom left corner, FALSE for vertical display on the bottom right corner.
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 bar in kilometers. If set to FALSE, no scale bar is displayed, if set to "auto" an automatic size is used (1/10 of the map width).
posscale  position of the scale, can be "bottomright", "bottomleft" or a vector of two coordinates (c(x, y))
frame  whether displaying a frame (TRUE) or not (FALSE).
legendBarsSymbols

north whether displaying a North arrow (TRUE) or not (FALSE).
south whether displaying a South arrow (TRUE) or not (FALSE).
extent sf object or Spatial*DataFrame; sets the extent of the frame to the one of a spatial object. (optional)
tabtitle size of the title box either a full banner (FALSE) or a "tab" (TRUE).
postitle position of the title, one of "left", "center", "right".

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

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq), col = "#D1914D", border = "white", bg = "#A6CAE0")
# Layout plot
layoutLayer()

plot(st_geometry(mtq), col = "#D1914D", border = "white", bg = "#A6CAE0")
# Layout plot
layoutLayer(title = "Martinique",
    author = paste0("cartography ", packageVersion("cartography")),
    tabtitle = TRUE, scale = 5, north = TRUE, frame = FALSE,
    theme = "sand.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, border = "black", lwd = 1,
    values.cex = 0.6, var, inches, col = "red", frame = FALSE,
    values.rnd = 0, style = "c")
Arguments

pos	position of the legend, one of "topleft", "top", "topright", "right", "bottomright", 
	"bottom", "bottomleft", "bottomleftextra", "left" or a vector of two coordinates 
	in map units (c(x, y)).

title.txt	title of the legend.

title.cex	size of the legend title.

cex	size of the legend. 2 means two times bigger.

border	color of the borders.

lwd	width of the borders.

values.cex	size of the values in the legend.

var	vector of values (at least min and max).

inches	height of the higher bar.

col	color of symbols.

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

values.rnd

titleNcex	size of the legend title.

cex	size of the legend. 2 means two times bigger.

Examples

library(sf)
mtq <- st_read(system.file("pkg/mrq.pkg", package="cartography"))
plot(st_geometry(mtq))
box()
legendBarsSymbols(pos = "topleft", title.txt = "Title of the legend",

title.cex = 0.8, values.cex = 0.6, cex = 1,

var = c(min(mtq$POP),max(mtq$POP)),
inches = 0.5,
col = "purple",
values.rnd=0, style ="e")

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",
border = "black", horiz = FALSE)
Arguments

pos position of the legend, one of "topleft", "top", "topright", "right", "bottomright", "bottom", "bottomleft", "bottomleftextra", "left" or a vector of two coordinates in map units (c(x, y)).

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) break-points. It is possible to use a vector of characters.

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'

border color of the box borders

horiz layout of legend, TRUE for horizontal layout

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq))
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")

legendChoro(pos = "topright", title.txt = "Title of the legend", title.cex = 0.8,
values.cex = 0.6,
breaks = c(0,"two","100","1 000","10,000", "1 Million"),
col = carto_pal(pal1 = "orange_pal",n1 = 5), values.rnd =2,
nodata = TRUE, nodata.txt = "No data available", frame = TRUE,
symbol="box")
Legend for Proportional Circles Maps

Description

Plot legend for proportional circles maps

Usage

```r
legendCirclesSymbols(pos = "topleft",
                     title.txt = "Title of the legend", title.cex = 0.8, cex = 1,
                     border = "black", lwd = 1, values.cex = 0.6, var, inches,
                     col = "#E84923", frame = FALSE, values.rnd = 0, style = "c")
```

Arguments

- `pos` position of the legend, one of "topleft", "top", "topright", "right", "bottomright", "bottom", "bottomleft", "bottomleftextra", "left" or a vector of two coordinates in map units (c(x, y)).
- `title.txt` title of the legend.
- `title.cex` size of the legend title.
- `cex` size of the legend. 2 means two times bigger.
- `border` color of the borders.
- `lwd` width of the borders.
- `values.cex` size of the values in the legend.
- `var` vector of values (at least min and max).
- `inches` radii of the biggest circle.
- `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.
- `style` either "c" or "e". The legend has two display styles, "c" stands for compact and "e" for extended.

Examples

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq))
box()

propSymbolsLayer(x = mtq, var = "POP",
                 inches = 0.2, legend.pos = "n")

legendCirclesSymbols(pos = "topleft", inches = 0.2,
                      title.txt = "Title of the legend", title.cex = 0.8,
                      border = "black", lwd = 1, values.cex = 0.6, var, inches,
                      col = "#E84923", frame = FALSE, values.rnd = 0, style = "c")
```
legendGradLines

Plot legend for graduated size lines maps.

Usage

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

Arguments

pos  position of the legend, one of "topleft", "top", "topright", "right", "bottomright",
    "bottom", "bottomleft", "bottomleftextra", "left" or a vector of two coordinates
    in map units (c(x, y)).
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.
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.
frame  whether to add a frame to the legend (TRUE) or not (FALSE).
Examples

```r
cartography

Examples

library(sf)
mtq <- st_read(system.file("pkg/mq.gpkg", package="cartography"))
plot(st_geometry(mtq))
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**

```r
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", "right", "bottomright", 
  "bottom", "bottomleft", "bottomleftextra", "left" or a vector of two coordinates in map units (c(x, y)).
- `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 (at least min and max).
- `lwd`  
  width of the larger line.
- `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

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq))
box()
legendPropLines(pos = "topleft", title.txt = "Title",
                 title.cex = 0.8, values.cex = 0.6, cex = 1,
                 var = c(10, 100),
                 lwd = 15,
                 col="red", frame=TRUE, values.rnd=0)
```

Description

Plot legends for double proportional triangles maps.

Usage

```r
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", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)).
- **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

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq))
box()
var <- runif(10, 0,100)
var2 <- runif(10, 0,100)
r <- sqrt(var)*1000
r2 <- sqrt(var2)*1000
legendPropTriangles(pos = "topright", var.txt = "population 1",
                    var2.txt = "population 2", title.txt="Population totale",
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**

```r
legendSquaresSymbols(pos = "topleft",
title.txt = "Title of the legend", title.cex = 0.8, cex = 1,
border = "black", lwd = 1, values.cex = 0.6, var, inches,
col = "red", frame = FALSE, values.rnd = 0, style = "c")
```

**Arguments**

- `pos`  position of the legend, one of "topleft", "top", "topright", "right", "bottomright", "bottom", "bottomleft", "bottomleftextra", "left" or a vector of two coordinates in map units (c(x, y)).
- `title.txt`  title of the legend.
- `title.cex`  size of the legend title.
- `cex`  size of the legend. 2 means two times bigger.
- `border`  color of the borders.
- `lwd`  width of the borders.
- `values.cex`  size of the values in the legend.
- `var`  vector of values (at least min and max).
- `inches`  length of the sides of the larger square.
- `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.
style    either "c" or "e". The legend has two display styles, "c" stands for compact and
         "e" for extended.

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq))
box()
legendSquaresSymbols(pos = "bottomright", title.txt = "Title of the legend",
                     title.cex = 0.8, values.cex = 0.6,
                     var = c(max(mtq$POP), min(mtq$POP)),
                     inches = 0.5,
                     col="red",
                     frame=TRUE, values.rnd=0, style ="c")

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", "right", "bottomright",
         "bottom", "bottomleft", "bottomleftextra", "left" or a vector of two coordinates
         in map units (c(x, y)).
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

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq))
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,
ndata = TRUE, nndata.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,
ndata = FALSE, frame = FALSE, symbol="line")

north              North Arrow

Description

Plot a north arrow.

Usage

north(pos = "topright", col = "grey20", south = FALSE)

Arguments

pos    position of the north arrow. It can be one of "topleft", "top", "topright", "right",
       "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in
       map units (c(x, y)).

col    arrow color.

south  plot a south arrow instead.

See Also

layoutLayer
Examples

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq))
box()
for (i in list("topleft", "top", "topright", "right", "bottomright",
               "bottom", "bottomleft", "left", c(746368, 1632993))){
  north(i, south = TRUE)
}
```

---

**nuts0.df**

**Nuts 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**

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://riate.cnrs.fr/?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)
**nuts1.spdf**

- **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**


**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://riate.cnrs.fr/?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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique nuts id (character)</td>
</tr>
<tr>
<td>emp2008</td>
<td>Active population in employment in 2008 (thousands persons) (numeric)</td>
</tr>
<tr>
<td>act2008</td>
<td>Active population in 2008 (thousands persons) (numeric)</td>
</tr>
<tr>
<td>unemp2008</td>
<td>Active population unemployed in 2008 (thousands persons) (numeric)</td>
</tr>
<tr>
<td>birth_2008</td>
<td>Number of birth in 2008 (live birth) (numeric)</td>
</tr>
<tr>
<td>death_2008</td>
<td>Number of death in 2008 (death) (numeric)</td>
</tr>
<tr>
<td>gdppps1999</td>
<td>Gross domestic product (Purchasing Power Standards) in 1999 (million euros) (numeric)</td>
</tr>
<tr>
<td>gdppps2008</td>
<td>Gross domestic product (Purchasing Power Standards) in 2008 (million euros) (numeric)</td>
</tr>
<tr>
<td>pop1999</td>
<td>Total population in 1999 (inhabitants) (numeric)</td>
</tr>
<tr>
<td>pop2008</td>
<td>Total population in 2008 (inhabitants) (numeric)</td>
</tr>
</tbody>
</table>

Source


---

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique nuts id (character)</td>
</tr>
</tbody>
</table>

Source

UMS RIATE - http://riate.cnrs.fr/?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**

---

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://riate.cnrs.fr/?page_id=153](http://riate.cnrs.fr/?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

```r
propLinkLayer(x, df, xid = NULL, dfid = NULL, 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, spdf,  
spdfid, spdfids, spdfide, dfids, dfide)
```

Arguments

- **x**: an sf object, a simple feature collection.
- **df**: a data frame that contains identifiers of starting and ending points and a variable.
- **xid**: identifier fields in x, character vector of length 2, default to the 2 first columns. (optional)
- **dfid**: identifier fields in df, character vector of length 2, default to the two first columns. (optional)
- **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", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)). 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.
propSymbolsChoroLayer

Description
Plot a proportional symbols layer with colors based on a quantitative data classification

Usage
propSymbolsChoroLayer(x, 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.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).
spdf defunct.
spdfid defunct.
spdfids defunct.
spdfide defunct.
dfids defunct.
dfide defunct.

Note
Unlike most of cartography functions, identifiers fields are mandatory.

See Also
gradLinkLayer, getLinkLayer, legendPropLines

Examples
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
mob <- read.csv(system.file("csv/mob.csv", package="cartography"))
# Create a link layer - work mobilities to Fort-de-France (97209)
mob.sf <- getLinkLayer(x = mtq, df = mob[mob$jid==97209], dfid = c("i", "j"))
# Plot the links - Work mobility
plot(st_geometry(mtq), col = "grey60", border = "grey20")
propLinkLayer(x = mob.sf, df = mob,
  maxlwd = 10,
  legend.pos = "topright",
  var = "fij",
  col = "#92000090", add = TRUE)
x an sf object, a simple feature collection. If x is used then spdf, df, spdfid and 
dfid are not.

spdf SpatialPointsDataFrame or SpatialPolygonsDataFrame; if spdf is a SpatialPoly-
gonsDataFrame 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. (op-
tional)

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".

col border color 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) break-
points (see choroLayer Details).

method a classification 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", "right", "bottomright", 
"bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, 
y)). If legend.var.pos is "n" then the legend is not plotted.
propSymbolsChoroLayer

- **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", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)). 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).
- **legend.var2.border**: color of boxes borders in the legend.
- **legend.var2.horiz**: whether to display the legend horizontally (TRUE) or not (FALSE).
- **add**: whether to add the layer to an existing plot (TRUE) or not (FALSE).

**See Also**

legendBarsSymbols, legendChoro, legendCirclesSymbols, legendSquaresSymbols, choroLayer, propSymbolsLayer

**Examples**

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq), col = "grey60", border = "white",
     lwd=0.4, bg = "lightsteelblue")
propSymbolsChoroLayer(x = mtq, var = "POP", var2 = "MED",
   col = carto.pal(pal1 = "blue.pal", n1 = 3,
                  pal2 = "red.pal", n2 = 3),
   inches = 0.2, method = "q6",
   border = "grey50", lwd = 1,
   legend.var.pos = "topright",
   legend.var2.pos = "left",
   legend.var2.values.rnd = -2,
   legend.var2.title.txt = "Median Income\n(in euros)",
   legend.var.title.txt = "Total Population",
   legend.var.style = "e")

# First layout
layoutLayer(title="Population and Wealth in Martinique, 2015")
```
propSymbolsLayer  

Proportional Symbols Layer

Description

Plot a proportional symbols layer.

Usage

propSymbolsLayer(x, spdf, df, spdfid = NULL, dfid = NULL, var, 
inches = 0.3, fixmax = NULL, symbols = "circle", col = "#E84923", 
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, breakval = NULL, col2)

Arguments

x  
an sf object, a simple feature collection. If x is used then spdf, df, spdfid and 
dfid are not.

spdf  
a SpatialPointsDataFrame or a SpatialPolygonsDataFrame; if spdf is a Spa-
tialPolygonsDataFrame 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. (op-
tional)

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

var  
named 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).

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

col  
color of symbols.

border  
color of symbols borders.

lwd  
width of symbols borders.

legend.pos  
position of the legend, one of "topleft", "top", "topright", "right", "bottomright", 
"bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, 
y)). 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.
propSymbolsTypoLayer

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).

breakval
defunct.

col2
defunct.

Details

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

See Also

legendBarsSymbols, legendCirclesSymbols, legendSquaresSymbols, propSymbolsChoroLayer, propSymbolsTypoLayer

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq))
propSymbolsLayer(x = mtq, var = "POP")

plot(st_geometry(mtq), col = "lightblue4", border = "lightblue3",
    bg = "lightblue1")
# Population plot on proportional symbols
propSymbolsLayer(x = mtq, var = "POP",
    symbols = "circle", col = "white",
    legend.pos = "right", border = "grey",
    legend.title.txt = "Total\nPopulation",
    legend.style = "c")
# Layout plot
layoutLayer(title = "Population Distribution in Martinique, 2015")
Usage

propSymbolsTypoLayer(x, 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)

Arguments

x an sf object, a simple feature collection. If x is used then spdf, df, spdfid and dfid are not.

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, one of "topleft", "top", "topright", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)).

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

legend.values.rnd number of decimal places of the values in the legend.
propSymbolsTypoLayer

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, one of "topleft", "top", "topright", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)).

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).

See Also

legendBarsSymbols, legendTypo, legendCirclesSymbols, legendSquaresSymbols, typoLayer, propSymbolsLayer

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
# Countries plot
plot(st_geometry(mtq), col = "lightblue4", border = "lightblue3", bg = "lightblue1")
# Population plot on proportional symbols
propSymbolsTypoLayer(x = mtq, var = "POP", var2 = "STATUS",
    symbols = "circle",
    col = c("aquamarine4", "yellow3", "wheat"),
    legend.var2.values.order = c("Prefecture",
        "Sub-prefecture",
        "Simple municipality"),
    legend.var.pos = "right", border = "grey",
    legend.var.title.txt = "Total\nPopulation")
layoutLayer(title = "Population Distribution in Martinique, 2015")
propTrianglesLayer  

**Double Proportional Triangle Layer**

**Description**

Plot a double proportional triangles layer.

**Usage**

```r
propTrianglesLayer(x, 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**

- **x**
  an sf object, a simple feature collection. If x is used then spdf, df, spdfid and dfid are not.
- **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.
smoothLayer

```
  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

```
library(sf)
mtq <- st_read(system.file("pkg/mqpkg", package="cartography"))
# Employed Active Population
mt$q$OCC <- mt$q$ACT-mt$q$CHOM
plot(st Geometry(mt$q), col = "lightblue", border = "lightblue",
     bg = "lightblue")
propTrianglesLayer(x = mt$q, var1 = "OCC", var2 = "CHOM",
                   col1="green",col2="red",k = 0.1)
l attLayer(title = "Active Population in Martinique, 2015")
```

---

**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(x, 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

x an sf object, a simple feature collection.
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 * mDis-
  tance) ^ (-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 func-
  tion equals 0.5.
beta numeric; impedance factor for the spatial interaction function.
resolution numeric; resolution of the output SpatialPointsDataFrame (in map units).
mask sf object or 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", "right", "bottomright",
  "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x,y)). If legend.pos is "n" then the legend is not plotted.
legend.title.txt title of the legend.
tilesLayer

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

```r
## Not run:
library(sf)
mtq <- st_read(system.file("pkg/mq.pkg", package="cartography"))
# Download the tiles, extent = Martinique
mtqOSM <- getTiles(x = mtq, type = "osm", crop = TRUE)
# Plot the tiles
tileLayer(mtqOSM)
# Plot countries
plot(st_geometry(mtq), add=TRUE)
txt <- "© OpenStreetMap contributors. Tiles style under CC BY-SA, www.openstreetmap.org/copyright"
mtext(text = txt, side = 1, adj = 0, cex = 0.7, font = 3)

## End(Not run)
```

twincities.df Twin Cities Dataset

Description

This dataset contains the number of international twinning agreements between 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
fi j number of agreements
Typology Layer

Description
Plot a typology layer.

Usage
```
typoLayer(x, 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
- **x**: an sf object, a simple feature collection. If x is used then spdf, df, spdfid and dfid are not.
- **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", "right", "bottomright", "bottom", "bottomleft", "left" or a vector of two coordinates in map units (c(x, y)). 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
    library(sf)
    mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
    typoLayer(x = mtq, var="STATUS",
              col = c("aquamarine4", "yellow3","wheat"),
              legend.values.order = c("Prefecture",
                                      "Sub-prefecture",
                                      "Simple municipality"),
              legend.pos = "topright",
              legend.title.txt = "Status")
    layoutLayer(title = "Municipality Status")
    
    world.spdf
    | World Background |

Description
    World background.

Format
    SpatialPolygonsDataFrame.

Source
    UMS RIATE - http://riate.cnrs.fr/?page_id=153
Index

barscale, 3, 6

carto.pal, 4, 6, 8, 12, 13
carto.pal.info, 5, 5, 12, 13, 30
cartography, 6
cartography-package (cartography), 6
choroLayer, 6, 7, 48, 49, 57
classIntervals, 16
coasts.pdf, 9
countries.pdf, 9
discLayer, 6, 10, 16, 22
display.carto.all, 5, 6, 12, 13
display.carto.pal, 5, 6, 12, 12
dotDensityLayer, 13

frame.pdf, 15

getBorders, 6, 10, 11, 15, 22
getBreaks, 7, 8, 10, 16
getFigDim, 18
getGridData, 19
getGridLayer, 7, 19
getOuterBorders, 16, 21
getPencilLayer, 22
getTiles, 7, 23, 58

gradLinkLayer, 6, 11, 21, 25, 28, 47
gradLinkTypoLayer, 6, 26

gridule.pdf, 28

invisible, 11, 57

labellayer, 7, 29, 31
layoutLayer, 3, 6, 29, 30, 40
legendBarsSymbols, 7, 31, 49, 51, 53
legendChoro, 7, 8, 32, 49
legendCirclesSymbols, 7, 34, 49, 51, 53
legendGradLines, 7, 11, 26, 28, 35
legendPropLines, 7, 36, 47
legendPropTriangles, 7, 37, 55

legendSquaresSymbols, 7, 38, 49, 51, 53
legendTypo, 7, 39, 53, 60

north, 6, 40
nuts0.df, 41
nuts0.spdf, 42
nuts1.df, 42
nuts1.spdf, 43
nuts2.df, 43
nuts2.spdf, 44
nuts3.df, 45

par, 18
points, 14
propLinkLayer, 6, 21, 26, 28, 46
propSymbolsChoroLayer, 6, 8, 47, 51
propSymbolsLayer, 6, 14, 48, 49, 50, 53
propSymbolsTypoLayer, 6, 51, 51, 60
propTrianglesLayer, 6, 54

quantile, 16
quickStewart, 55, 57

smoothLayer, 55
SpatialPosition, 55, 57
spsample, 14

text, 29
tilesLayer, 7, 24, 57
twincities.df, 58
typoLayer, 6, 53, 59, 60

world.pdf, 60