Package ‘cartography’

April 20, 2020

Title Thematic Cartography

Version 2.4.1

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, curl, graphics, methods, png, raster, Rcpp, rgeos,
        sf (>= 0.6-4), slippymath, sp (>= 1.2-4), stats, utils,
        grDevices

Suggests lwgeom, SpatialPosition, knitr, rmarkdown, rgdal, tinytest

LinkingTo Rcpp

VignetteBuilder knitr

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

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation yes

Author Timothée Giraud [cre, aut] (https://orcid.org/0000-0002-1932-3323),
        Nicolas Lambert [aut],
        Diego Hernangómez [ctb] (https://orcid.org/0000-0001-8457-4658),
        Ian Fellows [cph] (no overlap algorithm for labels, from wordcloud
        package)

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

Repository CRAN

Date/Publication 2020-04-20 12:30:02 UTC
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>barscale</td>
<td>3</td>
</tr>
<tr>
<td>carto.pal</td>
<td>4</td>
</tr>
<tr>
<td>choroLayer</td>
<td>5</td>
</tr>
<tr>
<td>discLayer</td>
<td>8</td>
</tr>
<tr>
<td>dotDensityLayer</td>
<td>10</td>
</tr>
<tr>
<td>getBorders</td>
<td>12</td>
</tr>
<tr>
<td>getBreaks</td>
<td>13</td>
</tr>
<tr>
<td>getFigDim</td>
<td>15</td>
</tr>
<tr>
<td>getGridLayer</td>
<td>16</td>
</tr>
<tr>
<td>getLinkLayer</td>
<td>17</td>
</tr>
<tr>
<td>getPencilLayer</td>
<td>19</td>
</tr>
<tr>
<td>getPngLayer</td>
<td>20</td>
</tr>
<tr>
<td>getTiles</td>
<td>21</td>
</tr>
<tr>
<td>ghostLayer</td>
<td>24</td>
</tr>
<tr>
<td>gradLinkLayer</td>
<td>24</td>
</tr>
<tr>
<td>gradLinkTypoLayer</td>
<td>26</td>
</tr>
<tr>
<td>hatchedLayer</td>
<td>29</td>
</tr>
<tr>
<td>labelLayer</td>
<td>30</td>
</tr>
<tr>
<td>layoutLayer</td>
<td>32</td>
</tr>
<tr>
<td>legendBarsSymbols</td>
<td>33</td>
</tr>
<tr>
<td>legendChoro</td>
<td>35</td>
</tr>
<tr>
<td>legendCirclesSymbols</td>
<td>36</td>
</tr>
<tr>
<td>legendGradLines</td>
<td>38</td>
</tr>
<tr>
<td>legendHatched</td>
<td>39</td>
</tr>
<tr>
<td>legendPropLines</td>
<td>40</td>
</tr>
<tr>
<td>legendPropTriangles</td>
<td>42</td>
</tr>
<tr>
<td>legendSquaresSymbols</td>
<td>43</td>
</tr>
<tr>
<td>legendTypo</td>
<td>44</td>
</tr>
<tr>
<td>legendWaffle</td>
<td>46</td>
</tr>
<tr>
<td>north</td>
<td>47</td>
</tr>
<tr>
<td>propLinkLayer</td>
<td>48</td>
</tr>
<tr>
<td>propSymbolsChoroLayer</td>
<td>50</td>
</tr>
<tr>
<td>propSymbolsLayer</td>
<td>52</td>
</tr>
<tr>
<td>propSymbolsTypoLayer</td>
<td>55</td>
</tr>
<tr>
<td>propTrianglesLayer</td>
<td>57</td>
</tr>
<tr>
<td>smoothLayer</td>
<td>59</td>
</tr>
<tr>
<td>tilesLayer</td>
<td>61</td>
</tr>
<tr>
<td>typoLayer</td>
<td>63</td>
</tr>
<tr>
<td>waffleLayer</td>
<td>65</td>
</tr>
<tr>
<td>wordcloudLayer</td>
<td>67</td>
</tr>
</tbody>
</table>
barscale

Scale Bar

Description

Plot a scale bar.

Usage

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

Arguments

size  size of the scale bar in units (default to km). 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.
unit  units used for the scale bar. Can be "mi" for miles, "m" for meters, or "km" for kilometers (default)

Note

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

See Also

layoutLayer

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq), col = "grey60", border = "grey20")
barscale(size = 5)
barscale(size = 5, lwd = 2, cex = .9, pos = c(714000, 1596000))
carto.pal  

Build Cartographic Palettes

Description

carto.pal builds sequential, diverging and qualitative color palettes. Diverging color palettes can be dissymmetric (different number of colors in each of the two gradients).
carto.pal.info displays the names of all color palettes.
display.carto.pal displays one color palette.
display.carto.all displays all the available color palettes.

Usage

carto.pal(
  pal1,
  n1,
  pal2 = NULL,
  n2 = NULL,
  middle = FALSE,
  transparency = FALSE
)
carto.pal.info()
display.carto.pal(name)
display.carto.all(n = 10)

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.
  name    name of the palette available in the package (see Details).
  n       number of colors in the gradient (up to 20).

Details


Qualitative palettes: "pastel.pal" or "multi.pal".
**Value**

carto.pal returns a vector of colors.
carto.pal.info returns a vector of color palettes names.

**References**

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

**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 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")
carto.pal.info()
display.carto.pal("orange.pal")
display.carto.all(8)
Usage

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
df 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 variable in spdf, default to the first column of the spdf data
          frame. (optional)
dfid      name of the identifier variable in df, default to the first column of df. (optional)
var       name of the numeric variable 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).
choroLayer

col
border
lwd
colNA
legend.pos
legend.title.txt
legend.title.cex
legend.values.cex
legend.values.rnd
legend.nodata
legend.frame
legend.border
legend.horiz
add

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")
```

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,
)```
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  name of the identifier variable in df, default to the first column of df. (optional)
var  name of the numeric variable 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`).
class  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.
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 \( \max(\text{value unit 1} / \text{value unit 2}, \text{value unit 2} / \text{value unit 1}) \).
The "abs" type of discontinuity is the result of \( \max(\text{value unit 1} - \text{value unit 2}, \text{value unit 2} - \text{value unit 1}) \).
Value

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

See Also

getBorders, gradLinkLayer, legendGradLines

Examples

```r
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\n(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\n(absolute difference)",
legend.pos = "bottomleft", add=TRUE)
```

---

**dotDensityLayer**

**Dot Density Layer**

**Description**

Plot a dot density layer.

**Usage**

```r
dotDensityLayer(
x, 
spdf, 
df, 
spdfid = NULL, 
dfid = NULL, 
var, 
n = NULL, 
pch = 1, 
cex = 0.15, 
type = "random", 
col = "black", 
iter, 
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 name of the identifier variable in spdf, default to the first column of the spdf data frame. (optional)
dfid name of the identifier variable in df, default to the first column of df. (optional)
var name of the numeric variable to plot.
n one dot on the map represents n (in var units).
pch symbol to use: points.
cex size of the symbols

Details

The type parameters is defined within the st_sample function.

See Also

propSymbolsLayer
### Examples

```r
## Not run:
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
plot(st_geometry(mtq), col = "#B8704D50")
dotDensityLayer(x = mtq, var="POP", pch=20, col = "red4", n = 200)
layoutLayer(title = "Population Distribution in Martinique, 2015")

## End(Not run)
```

### getBorders

#### Extract Polygons Borders

---

**Description**

Extract borders between polygons.

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

**Usage**

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

**Arguments**

- `x`: an sf object, a simple feature collection or a SpatialPolygonsDataFrame.
- `id`: name of the identifier variable in `x`, 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)
- `res`: resolution of the grid used to compute outer borders (in x units). A high resolution will give more detailed borders. (optional)
- `width`: maximum distance between used to compute outer borders (in x units). A higher width will build borders between units that are farther apart. (optional)

**Value**

An sf object (MULTILINESTRING) of borders is returned. This object has three id variables: 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`. 
getBreaks

See Also
discLayer

Examples

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)

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)

getBreaks

Classification

Description

A function to classify continuous 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 classification method; one of "fixed", "sd", "equal", "pretty", "quantile", "kmeans",
        "hclust", "bclust", "fisher", "jenks", "dpih", "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).
...      further arguments of classIntervals.
Details

"fixed", "sd", "equal", "pretty", "quantile", "kmeans", "hclust", "bclust", "fisher", "jenks" and "dpih" are classIntervals methods. You may need to pass additional arguments for some of them.

Jenks ("jenks" method) and Fisher-Jenks ("fisher" method) 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 of classIntervals + "arith", "em", "q6", "geom" and "msd" methods.

See Also

classIntervals

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", 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 = "#F0D9F9")
rug(var)
```r
med <- median(var)
abline(v = med, col = "blue", lwd = 3)

# Pretty breaks
breaks <- getBreaks(v = var, nclass = 4, method = "pretty")
hist(var, probability = TRUE, breaks = breaks, col = "#F0D9F9", axes = FALSE)
rug(var)
axis(1, at = breaks)
axis(2)
abline(v = med, col = "blue", lwd = 6)

# kmeans method
breaks <- getBreaks(v = var, nclass = 4, method = "kmeans")
hist(var, probability = TRUE, breaks = breaks, col = "#F0D9F9")
rug(var)
abline(v = med, col = "blue", lwd = 6)

# 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 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**

```r
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.
**getGridLayer**

Build a Regular Grid Layer

**Description**

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

**Parameters**

- **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
ggplot(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("Madinina")
dev.off()

## PDF export
# get figure dimension
sizes <- getFigDim(x = mtq, width = 450, mar = c(1,1,2.2,1))
# export the map
ggplot(filename = "mtq.pdf", width = sizes[1]/72, height = sizes[2]/72)
par(mar = c(1,1,2.2,1))
plot(st_geometry(mtq), col = "#D1914D", border = "white", bg = "#A6CAE0")
title("Madinina")
dev.off()
## End(Not run)
```
getGridLayer(x, cellsize, type = "regular", var, spdf, spdfid = NULL)

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 variable(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$POPDENS, method = "geom", 5)
cols <- carto.pal(pal1 = "taupe.pal", n1 = 5)
opar <- par(mfrow = c(1,2), mar = c(0,0,0,0))
choroLayer(x = mtq, var = "POPDENS", breaks = bks,
    border = "burlywood3", col = cols,
    legend.pos = "topright", legend.values.rnd = 0,
    legend.title.txt = "Population density")
mygrid <- getGridLayer(x = mtq, cellsize = 3e7,
    type = "hexagonal", var = "POP")
## conversion from square meter to square kilometers
mygrid$POPDENSG <- 1e6 * mygrid$POP / mygrid$gridarea
choroLayer(x = mygrid, var = "POPDENSG", breaks = bks,
    border = "burlywood3", col = cols,
    legend.pos = "n", legend.values.rnd = 1,
    legend.title.txt = "Population density")
par(opar)
```

getLinkLayer

Create a Links Layer from a Data Frame of Links.

Description

Create a links layer from a data frame of links.
getLinkLayer

Usage

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

Arguments

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

xid name of the identifier variable in x, default to the first column (optional)

df a data frame that contains identifiers of starting and ending points.

dfid names of the identifier variables 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 variables (origins and destinations).

See Also

gradLinkLayer, propLinkLayer

Examples

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 links1
plot(st_geometry(mtq), col = "grey")
plot(st_geometry(mob.sf), col = "red4", lwd = 2, 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**

```r
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")
```
**getPngLayer**

**Description**

Get a RasterBrick from a `.png` image cut using the shape of a spatial object. The `.png` file could be either a local file or extracted from a given url.

**Usage**

```r
getPngLayer(
  x,
  pngpath,
  align = "center",
  margin = 0,
  crop = FALSE,
  mask = TRUE,
  inverse = FALSE,
  dwmode = "curl",
  ...
)
```

**Arguments**

- **x** an `sf` object, a simple feature collection (POLYGON or MULTIPOLYGON) or a tile (see `getTiles`).
- **pngpath** local path or url of a `.png` file.
- **align** set how the `.png` file should be fitted within `x`. Possible values are 'left', 'right', 'top', 'bottom' or 'center'.
- **margin** inner margin, zooms out the `.png` over `x`. If 0 then `.png` is completely zoomed over `x`.
- **crop** TRUE if results should be cropped to the specified `x` extent.
- **mask** TRUE if the result should be masked to `x`.
- **inverse** logical. If FALSE, overlapped areas of `x` on `pngpath` are extracted, otherwise non-overlapping areas are returned. See `mask`.
- **dwmode** Set the download mode. It could be 'base' for `download.file` or 'curl' for `curl_download`.
- **...** additional arguments for downloading the file. See `download.file` or `curl_download`.

**Details**

The effect of `align` would differ depending of the aspect ratio of `x` and `pngpath`. To obtain a fitted tile from `pngpath` given that `x` is the tile to fit, set `margin = 0`, `crop = TRUE`.
getTiles

Value

A RasterBrick object is returned.

Note

The accuracy of the final plot would depend on the quality of the .png file, the scale of x and the resolution setup of the graphic device. Exporting to svg is highly recommended.

Author(s)

dieghernan, https://github.com/dieghernan/

See Also

pngLayer

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package = "cartography"))
# Local file
dirpng <- system.file("img/LogoMartinique.png", package = "cartography")
mask <- getTiles(mtq, dirpng)

## Not run:
# Remote file
urlpng <- "https://i.imgur.com/gePiDvB.png"
masksea <- getTiles(mtq, urlpng, mode = "wb", inverse = TRUE)

## End(Not run)

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 = "OpenStreetMap",
  zoom = NULL,
  crop = FALSE,
  verbose = FALSE,
  apikey = NA,
getTiles

cachedir = FALSE,
forceDownload = FALSE
)

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. See Details for providers. For other
sources use a list: type = list(src = "name of the source", q = "tiles address", sub
= "subdomains", cit = "how to cite the tiles"). See Examples.

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.
If x is an sf object with one POINT, crop is set to FALSE.

verbose
if TRUE, tiles filepaths, zoom level and citation are displayed.

apikey
Needed for Thunderforest maps.

cachedir
name of a directory used to cache tiles. If TRUE, places a `tile.cache` folder in
the working directory. If FALSE, tiles are not cached.

forceDownload
if TRUE, cached tiles are downloaded again.

Details

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

Full list of providers:

- `'OpenStreetMap'` (or `osm`)  
- `'OpenStreetMap.DE'`  
- `'OpenStreetMap.France'`  
- `'OpenStreetMap.HOT'` (or `hotstyle`)  
- `'OpenMapSurfer'`  
- `'OpenMapSurfer.Roads'`  
- `'OpenMapSurfer.Hybrid'`  
- `'OpenMapSurfer.AdminBounds'`  
- `'OpenMapSurfer.ElementsAtRisk'`  
- `'CartoDB'`  
- `'CartoDB.Positron'` (or `cartolight`)  
- `'CartoDB.PositronNoLabels'`  
- `'CartoDB.PositronOnlyLabels'`  
- `'CartoDB.DarkMatter'` (or `cartodark`)  
- `'CartoDB.DarkMatterNoLabels'`  
- `'CartoDB.DarkMatterOnlyLabels'`  
- `'CartoDB.Voyager'`  
- `'CartoDB.VoyagerNoLabels'`  
- `'Stamen'` (or `stamenbw`)  
- `'Stamen.Toner'`  
- `'Stamen.TonerBackground'`  
- `'Stamen.TonerHybrid'`  
- `'Stamen.TonerLines'`  
- `'Stamen.TonerLabels'`  
- `'Stamen.TonerLite'`  
- `'Stamen.Watercolor'` (or `stamenwatercolor`)  
- `'Stamen.Terrain'`  
- `'Stamen.TerrainBackground'`  
- `'Stamen.TerrainLabels'`  
- `'Thunderforest'`  
- `'Thunderforest.Transport'`  
- `'Thunderforest.TransportDark'`  
- `'Thunderforest.SpinalMap'`  
- `'Thunderforest.Landscape'`  
- `'Thunderforest.Outdoors'`  
- `'Thunderforest.Pioneer'`  
- `'Hydda'`  
- `'Hydda.Full'`  
- `'Hydda.Base'`  
- `'Hydda.RoadsAndLabels'`  
- `'HikeBike'` (or `hikebike`)  
- `'HikeBike.HikeBike'`  
- `'OpenTopoMap'` (or `opentopomap`)
Value

A RasterBrick is returned.

References

https://leaflet-extras.github.io/leaflet-providers/preview/

See Also

tilesLayer

Examples

```r
## Not run:
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", 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 <- paste0("\u00A9 OpenStreetMap contributors.,
" Tiles style under CC BY-SA, www.openstreetmap.org/copyright")
mtext(text = txt, side = 1, adj = 0, cex = 0.7, font = 3)
# Download esri tiles
fullserver = paste("https://server.arcgisonline.com/ArcGIS/rest/services",
"Specialty/DeLorme_World_Base_Map/MapServer",
"tile/{z}/{y}/{x}.jpg",
sep = "/")
typeosm <- list(
  src = 'esri',
  q = fullserver,
  sub = NA,
  cit = 'Tiles; Esri; Copyright: 2012 DeLorme'
)
mtqESRI <- getTiles(x = mtq, type = typeosm, crop = TRUE, verbose = T, zoom = 10)
# Plot the tiles
tilesLayer(mtqESRI)
txt <- typeosm$cit
mtext(text = txt, side = 1, adj = 0, cex = 0.6, font = 3)
## End(Not run)
```
**ghostLayer**

*Plot a Ghost Layer*

**Description**

Plot an invisible layer with the extent of a spatial object.

**Usage**

```r
ghostLayer(x, bg)
```

**Arguments**

- `x`: an sf object, a simple feature collection or a Spatial*DataFrame.
- `bg`: background color.

**Examples**

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
target <- mtq[30,]
ghostLayer(target, bg = "lightblue")
plot(st_geometry(mtq), add = TRUE, col = "gold2")
plot(st_geometry(target), add = TRUE, col = "red")
# overly complicated label placement trick:
labelLayer(x = suppressWarnings(st_intersection(mtq, st_buffer(target, 2000))),
txt = "LIBGEO", halo = TRUE, cex = .9, r = .14, font = 2,
bg = "grey20", col = "white")
```

**gradLinkLayer**

*Graduated Links Layer*

**Description**

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

**Usage**

```r
gradLinkLayer(
  x,
  df,
  xid = NULL,
  dfid = NULL,
  var,
  breaks = getBreaks(v = df[, var], nclass = 4, method = "quantile"),
  lwd = c(1, 2, 4, 6),
)```
gradLinkLayer

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 names of the identifier variables in x, character vector of length 2, default to the 2 first columns. (optional)

dfid names of the identifier variables 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).

spdf defunct.

spdfid defunct.
gradLinkTypoLayer

spfids   defunct.
spfide   defunct.
dfids    defunct.
dfide    defunct.

Note

Unlike most of cartography functions, identifiers fields are mandatory.

See Also

gLinkLayer, propLinkLayer, legendGradLines

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$j==97209,], dfid = c("i", "j"))
# Plot the links - Work mobility
plot(st_geometry(mtq), col = "grey60", border = "grey20")
gradLinkLayer(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)

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 depend on a discrete variable of categories.

Usage

gradLinkTypoLayer(
  x,
  df,
  xid = NULL,
  dfid = NULL,
  var,
  breaks = getBreaks(v = df[, var], nclass = 4, method = "quantile"),
  lwd = c(1, 2, 4, 6),
  var2,
gradLinkTypoLayer

\begin{verbatim}
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",
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 names of the identifier variables in x, character vector of length 2, default to the 2 first columns. (optional)
dfid names of the identifier variables 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).
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)).
\end{verbatim}
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 variables are mandatory.

See Also
getLinkLayer, 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$j %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)
hatchedLayer

**hatchedLayer**

**Hatched Layer**

**Description**

Plot a hatched layer with several different patterns. Suitable for b/w print maps.

**Usage**

```r
hatchedLayer(x, pattern = "dot", density = 1, txt = "a", ...)
```

**Arguments**

- **x**: an sf object, a simple feature collection. It should be either a POLYGON or a MULTIPOLYGON.
- **pattern**: Desired pattern to use for hatching. Possible values are:
  - Dots: "dot", "text"
  - Lines: "diamond", "grid", "hexagon", "horizontal", "vertical", "zigzag", "left2right", "right2left"
- **density**: of the grid. By default the function uses a grid with a minimum of 10 cells on the shortest dimension of the bounding box. Additionally, it is possible to pass a `cellsize` value that would feed the `st_make_grid` underlying function.
- **txt**: for the "text" pattern, that should be a character.
- **...**: Additional graphic parameters (see Details).

**Details**

Possible values are:

<table>
<thead>
<tr>
<th>Pattern</th>
<th>add</th>
<th>col</th>
<th>bg</th>
<th>cex</th>
<th>pch</th>
<th>lwd</th>
<th>lty</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;dot&quot;</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;text&quot;</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Patterns</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Value**

When passing `mode=’sfc’` an 'sf' object (either MULTILINESTRING or MULTIPOINT) is returned.

**Author(s)**

dieghernan, https://github.com/dieghernan/

**See Also**

legendHatched
Examples

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package = "cartography"))
par(mar=c(1,1,1,1))
hatchedLayer(mtq, "dot")
title("dot")
plot(st_geometry(mtq), border = NA, col="grey80")
hatchedLayer(mtq, "text", txt = "Y", add=TRUE)
title("text")
hatchedLayer(mtq, "diamond", density = 0.5)
plot(st_union(st_geometry(mtq)), add = TRUE)
title("diamond")
hatchedLayer(mtq, "grid", lwd = 1.5)
title("grid")
hatchedLayer(mtq, "hexagon", col = "blue")
title("hexagon")
hatchedLayer(mtq, "horizontal", lty = 5)
title("horizontal")
hatchedLayer(mtq, "vertical")
title("vertical")
hatchedLayer(mtq, "left2right")
title("left2right")
hatchedLayer(mtq, "right2left")
title("right2left")
hatchedLayer(mtq, "zigzag", cellsize=5000)
title("zigzag")
hatchedLayer(mtq, "circle")
```

labelLayer

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Put labels on a map.</td>
</tr>
</tbody>
</table>

Usage

```r
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, df, dfid and spdfid are not used.

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
name of the identifier variable in spdf, default to the first column of the spdf data frame. (optional)

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

txt
labels variable.

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)
Description

Plot a layout layer.

Usage

```r
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))
**legendBarsSymbols**

- **frame**: whether displaying a frame (TRUE) or not (FALSE).
- **north**: whether displaying a North arrow (TRUE) or not (FALSE).
- **south**: whether displaying a South arrow (TRUE) or not (FALSE).
- **extent**: `sf` object or `SpatialDataFrame`; 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**

```r
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**

```r
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  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()
legendBarsSymbols(pos = "topleft", title.txt = "Title of the legend",


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.
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,
)```

Examples

```r
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")
```
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,
var = c(min(mtq$POP), max(mtq$POP)))
legendCirclesSymbols(pos = "left",
var = c(min(mtq$POP), max(mtq$POP)),
inches = 0.2, style = "e")
legendCirclesSymbols(pos = "bottomleft",
var = c(600, 12000, 40000, max(mtq$POP)),
inches = 0.2, style = "c")
legendCirclesSymbols(pos = "topright", cex = 2,
var = c(600, 30000, max(mtq$POP)),
inches = 0.2, style = "e", frame = TRUE)
legendCirclesSymbols(pos = c(736164.4, 1596658),
var = c(min(mtq$POP), max(mtq$POP)),
inches = 0.2, frame = TRUE)
```
Legend for Graduated Size Lines Maps

Description

Plot legend for graduated size lines maps.

Usage

```r
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
library(sf)
mtq <- st_read(system.file("gpkg/mtq.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)

Description
Plot legend for hatched maps.

Usage
legendHatched(
pos = "topleft",
title.txt = "Title of the legend",
title.cex = 0.8,
values.cex = 0.6,
categ,
patterns,
ptrn.bg = "white",
ptrn.text = "X",
dot.cex = 0.5,
text.cex = 0.5,
cex = 1,
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.
values.cex size of the values in the legend.
categ vector of categories.
patterns vector of patterns to be created for each element on categ, see hatchedLayer.
ptrn.bg background of the legend box for each categ.
ptrn.text text to be used for each categ="text", as a single value or a vector.
dot.cex cex of each patterns = "dot" categories, as a single value or a vector.
text.cex text size of each patterns = "text" categories, as a single value or a vector.
cex size of the legend. 2 means two times bigger.
frame whether to add a frame to the legend (TRUE) or not (FALSE).
... optional graphical parameters, see details on hatchedLayer

Note
It is also possible to create solid legends, by setting col and ptrn.bg to the same color. Parameters would honour the order of the categ variable.

Author(s)
dieghernan, https://github.com/dieghernan/

See Also
hatchedLayer, legendTypo

Examples
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package = "cartography"))
typoLayer(mtq, var = "STATUS", legend.pos = "n", legend.values.order = c("Prefecture","Sub-prefecture", "Simple municipality"), col = c("grey10", "grey50", "grey80"), border = NA)
mtq$Patts <- cut(mtq$MED, c(-Inf, 15700, Inf), labels=FALSE)
hatchedLayer(mtq[mtq$Patts == 1, ], "left2right", density = 2, col = "white", add = TRUE, pch = 3, cex = 0.6)
hatchedLayer(mtq[mtq$Patts == 2, ], "left2right", density = 4, col = "white", add = TRUE)
legendHatched(pos = "bottomleft", cex = 1.5, values.cex = 0.8, title.txt = "Median Income\n(in thousand of euros)", categ = c("11.9 - 15.7","14.7 - 21.8", "Prefecture","Sub-prefecture", "Simple municipality"), patterns = c("left2right"), density = c(1, 2), col = c(rep("black", 2), "grey10", "grey50", "grey80"), ptrn.bg = c(rep("white", 2), "grey10", "grey50", "grey80"), pch = 3)
plot(st_geometry(st_union(mtq)), add = TRUE)
legendPropLines

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", "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 twice as big.
- **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

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)
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", "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.
**legendSquaresSymbols**

```
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\nthe 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")

Legend for Typology Maps

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,
Description

Plot legend for typology maps.

Usage

legendWaffle(
  pos = "topleft",
  title.txt = "Title of the legend",
  title.cex = 0.8,
  values.cex = 0.6,
  categ,
  cex = 1,
  cell.txt = "1 cell = ...",
  col,
  cell.size,
  border = "white",
  lwd = 0.2,
  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.
- values.cex: size of the values in the legend.
- categ: vector of categories.
- cex: size of the legend. 2 means two times bigger.
- cell.txt: label for cell values.
- col: a vector of colors.
- cell.size: size of the cell
north

border color of the cells borders.
lwd width of the cells borders
frame whether to add a frame to the legend (TRUE) or not (FALSE).

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")
legendWaffle(categ = someLabels, col = someColors, cell.size = 750)

Description

Plot a north arrow.

Usage

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

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.
x sf or sp object used to correct the north azimuth

See Also

layoutLayer

Examples

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)
}
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,
dfidx = 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**: names of the identifier variables in x, character vector of length 2, default to the 2 first columns. (optional)
- **dfid**: names of the identifier variables 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.
propLinkLayer

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).

spdf
defunct.

spdfid
defunct.

spdfids
defunct.

spdfide
defunct.

dfids
defunct.

dfide
defunct.

Note

Unlike most of cartography functions, identifiers variables 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$j==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)
propSymbolsChoroLayer  Proportional and Choropleth Symbols Layer

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.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,  
  legend.var2.border = "black",  
  legend.var2.horiz = FALSE,  
  add = TRUE
)

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

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 name of the identifier variable in spdf, default to the first column of the spdf data frame. (optional)

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

var name of the numeric variable used 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 variable used to plot the symbols 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.

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.
propSymbolsLayer

Description

Plot a proportional symbols layer.

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 = "lightsteelblue1")
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

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              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).
  symbols          type of symbols, one of "circle", "square" or "bar".
  col              color of symbols.
  border           color of symbols borders.
  lwd              width of symbols borders.
propSymbolsLayer

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.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")
propSymbolsTypoLayer  Proportional Symbols Typo Layer

Description

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

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.
propSymbolsTypoLayer

spdfid  name of the identifier variable in spdf, default to the first column of the spdf data frame. (optional)
dfid  name of the identifier variable in df, default to the first column of df. (optional)
var  name of the numeric variable used 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) variable used to plot the symbols colors.
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.
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).
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,
)```

**See Also**

`legendBarsSymbols`, `legendTypo`, `legendCirclesSymbols`, `legendSquaresSymbols`, `typoLayer`, `propSymbolsLayer`

**Examples**

```r
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

```r
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**: name of the identifier variable in spdf, default to the first column of the spdf data frame. (optional)
- **dfid**: name of the identifier variable in df, default to the first column of df. (optional)
- **var1**: name of the first numeric variable to plot, positive values only (top triangle).
- **col1**: color of top triangles.
- **var2**: name of the second numeric variable 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).
smoothLayer

See Also

legendPropTriangles

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
# Employed Active Population
mtq$OCC <- mtq$ACT-mtq$CHOM
plot(st_geometry(mtq), col = "lightblue4", border = "lightblue3",
     bg = "lightblue1")
propTrianglesLayer(x = mtq, var1 = "OCC", var2 = "CHOM",
                   col1="green4",col2="red4",k = 0.1)
layoutLayer(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,
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 variable in spdf, default to the first column of the spdf data frame. (optional)
- **dfid**: name of the identifier variable in df, default to the first column of df. (optional)
- **var**: name of the numeric variable used to compute potentials.
- **var2**: name of the numeric variable used to compute potentials. This variable 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: \((1 + \alpha \times mDistance)^{-\beta}\). If "exponential" the interaction is defined as: \(exp(-\alpha \times mDistance ^ \beta)\). 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**: 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.
tilesLayer

```r
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 sf object (MULTIPOLYGONs) is returned (see quickStewart).

**See Also**

quickStewart, SpatialPosition, choroLayer

**Examples**

```r
library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package="cartography"))
smoothLayer(x = mtq, var = 'POP',
  span = 4000, beta = 2,
  mask = mtq, border = NA,
  col = carto.pal(pal1 = 'wine.pal', n1 = 8),
  legend.title.txt = "Population\nPotential",
  legend.pos = "topright", legend.values.rnd = 0)
propSymbolsLayer(x = mtq, var = "POP", legend.pos = c(690000, 1599950),
  legend.title.txt = "Population 2015",
  col = NA, border = "#ffffff50")
layoutLayer(title = "Actual and Potential Popultation in Martinique")
```

---

**tilesLayer**

*Plot a Raster Object*

**Description**

Plot a raster object over a map. It can be used to plot tiles from getTiles or images from getPngLayer.
Usage

```r
tilesLayer(x, add = FALSE, 
pngLayer(x, add = FALSE, 
```

Arguments

- **x**: a RasterBrick object; `getPngLayer` and `getTiles` functions output these objects.
- **add**: whether to add the layer to an existing plot (TRUE) or not (FALSE).
- **...**: bgalpha, interpolate, or other arguments passed to `plotRGB`

Note

This function is a wrapper for `plotRGB` from the raster package. The accuracy of the final plot depends on the quality of the ".png" file, the scale of `x` and the resolution setup of the graphic device.

Author(s)

dieghernan, [https://github.com/dieghernan/](https://github.com/dieghernan/)

See Also

`getPngLayer`, `getTiles`

Examples

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

## Not run:
# 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)

# Local image
dirpng <- system.file("img/LogoMartinique.png", package = "cartography")
mask <- getPngLayer(mtq, dirpng, crop = TRUE, margin = 0.5)
par(mar = c(0,0,0,0))
ghostLayer(mtq)
pngLayer(mask, add = TRUE)

## Not run:
```
# Remote image
urlpng = "https://i.imgur.com/gePiDvB.png"
masksea <- getPngLayer(mtq, urlpng, mode = "wb", inverse = TRUE, margin = 0.5)
#Combine
par(mar = c(0,0,0,0))
ghostLayer(mtq)
pngLayer(mask, add = TRUE)
pngLayer(masksea, add = TRUE)
plot(st_geometry(mtq), border="orange", add=TRUE)
## End(Not run)

typoLayer

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  name of the identifier variable in spdf, default to the first column of the spdf data frame. (optional)

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

var  name of the variable 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")
waffleLayer

**Description**

Plot a waffle layer.

**Usage**

```r
waffleLayer(
  x, 
  var, 
  cellvalue, 
  cellsize, 
  cellrnd = "ceiling", 
  celltxt = paste0("\1 cell = ", cellvalue), 
  labels, 
  ncols, 
  col, 
  border = "white", 
  lwd = 0.2, 
  legend.pos = "bottomleft", 
  legend.title.txt = "legend title", 
  legend.title.cex = 0.8, 
  legend.values.cex = 0.6, 
  legend.frame = FALSE, 
  add = TRUE 
)
```

**Arguments**

- **x**: an sf object, a simple feature collection.
- **var**: names of the numeric variable to plot.
- **cellvalue**: value of a single cell. Original values are rounded, using `cellrnd` method, to be expressed as multiple of `cellvalue`.
- **cellsize**: size of single cell, in map units.
- **cellrnd**: rounding method, one of "ceiling", "floor", "round".
- **celltxt**: text that appears under the legend.
- **labels**: names that will appear in the legend.
- **ncols**: number of columns of the waffles
- **col**: a vector of colors.
- **border**: color of the cells borders.
- **lwd**: cells borders width.
waffleLayer

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.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).

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package = "cartography"),
             quiet = TRUE)
# number of employed persons
mtq$EMP <- mtq$ACT - mtq$CHOM

plot(st_geometry(mtq),
col = "#f2efe9",
border = "#b38e43",
lwd = 0.5)
waffleLayer(
x = mtq,
var = c("EMP", "CHOM"),
cellvalue = 100,
cellsze = 400,
cellrnd = "ceiling",
celltnt = "1 cell represents 100 persons",
labels = c("Employed", "Unemployed"),
ncols = 6,
col = c("tomato1", "lightblue"),
border = "#f2efe9",
legend.pos = "topright",
legend.title.cex = 1,
legend.title.txt = "Active Population",
legend.values.cex = 0.8,
add = TRUE
)

layoutLayer(
title = "Structure of the Active Population",
col = "tomato4",
tabtitle = TRUE,
scale = FALSE,
sources = paste0("cartography ", packageVersion("cartography")),
author = "Sources: Insee and IGN, 2018",
)
Description

Plot a word cloud adjusted to an sf object.

Usage

wordcloudLayer(
  x,  
  txt, 
  freq, 
  max.words = NULL, 
  cex.maxmin = c(1, 0.5), 
  rot.per = 0.1, 
  col = NULL, 
  fittopol = FALSE, 
  use.rank = FALSE, 
  add = FALSE, 
  breaks = NULL, 
  method = "quantile", 
  nclass = NULL 
)

Arguments

  x  an sf object, a simple feature collection (POLYGON or MULTIPOLYGON).
  txt labels variable.
  freq frequencies of txt.
  max.words Maximum number of words to be plotted. least frequent terms dropped
  cex.maxmin integer (for same size in all txt) or vector of length 2 indicating the range of the size of the words.
  rot.per proportion words with 90 degree rotation
  col color or vector of colors words from least to most frequent
  fittopol logical. If true would override rot.per for some elements of x
  use.rank logical. If true rank of frequencies is used instead of real frequencies.
  add whether to add the layer to an existing plot (TRUE) or not (FALSE)
  breaks, method, nclass additional arguments for adjusting the colors of txt, see choroLayer.

Author(s)

dieghernan, https://github.com/dieghernan/
References


R package version 2.6. https://CRAN.R-project.org/package=wordcloud

See Also

choroLayer, legendChoro

Examples

library(sf)
mtq <- st_read(system.file("gpkg/mtq.gpkg", package = "cartography"))
par(mar=c(0,0,0,0))
plot(st_geometry(mtq),
    col = "white",
    bg = "grey95",
    border = NA)
wordcloudLayer(
    x = mtq,
    txt = "LIBGEO",
    freq = "POP",
    add = TRUE,
    nclass = 5
)
legendChoro(
    title.txt = "Population",
    breaks = getBreaks(mtq$POP, nclass = 5, method = "quantile"),
    col = carto.pal("blue.pal", 5),
    nodata = FALSE
)
Index

barscale, 3

carto.pal, 4, 7
carto.pal.info, 32
cellsize, 29
choroLayer, 5, 51, 52, 61, 67, 68
classIntervals, 13, 14
curl_download, 20
discLayer, 8, 13
display.carto.all (carto.pal), 4
display.carto.pal (carto.pal), 4
dotDensityLayer, 10
download.file, 20
getBorders, 8–10, 12
getBreaks, 6, 7, 9, 13
getFigDim, 15
getGridLayer, 16
getLinkLayer, 17, 26, 28, 49
getOuterBorders (getBorders), 12
getPencilLayer, 19
getPngLayer, 20, 62
getTiles, 20, 21, 62
ghostLayer, 24
gradLinkLayer, 10, 18, 24, 28, 49
gradLinkTypoLayer, 26
hatchedLayer, 29, 39, 40
invisible, 10, 61
labelLayer, 30, 33
layoutLayer, 3, 31, 32, 47
legendBarsSymbols, 33, 52, 54, 57
legendChoro, 7, 35, 52, 68
legendCirclesSymbols, 36, 52, 54, 57
legendGradLines, 10, 26, 28, 38
legendHatched, 29, 39
legendPropLines, 40, 49
legendPropTriangles, 42, 59
legendSquaresSymbols, 43, 52, 54, 57
legendTypo, 40, 44, 57, 64
legendWaffle, 46
mask, 20
north, 47
par, 16
plotRGB, 62
pngLayer, 21
pngLayer (tilesLayer), 61
points, 11
propLinkLayer, 18, 26, 28, 48
propSymbolsChoroLayer, 7, 50, 54
propSymbolsLayer, 11, 51, 52, 57
propSymbolsTypoLayer, 54, 55, 64
propTrianglesLayer, 57
quantile, 14
quickStewart, 59, 61
smoothLayer, 59
SpatialPosition, 59, 61
st_make_grid, 29
st_sample, 11
text, 31
tilesLayer, 23, 61
typoLayer, 57, 63, 64
waffleLayer, 65
wordcloudLayer, 67