Package ‘mapmisc’

February 22, 2015

Type Package
Title Utilities for Producing Maps
Version 1.1.3
Date 2015-02-16
Depends sp, raster, R (>= 3.0.0)
Suggests RColorBrewer, geonames, classInt, rgdal, knitr
Author Patrick Brown <patrick.brown@utoronto.ca> [aut, cre], Barry Rowlingson [ctb]
Maintainer Patrick Brown <patrick.brown@utoronto.ca>
Description A minimal, light-weight set of tools for producing nice looking maps in R.
License GPL
VignetteBuilder knitr
Repository CRAN
Repository/R-Forge/Project diseasemapping
Repository/R-Forge/Revision 1011
Repository/R-Forge/DateTimeStamp 2015-02-17 02:25:47
Date/Publication 2015-02-22 00:18:29
NeedsCompilation no

R topics documented:

colourScale ................................................................. 2
GNcities ................................................................. 4
legendBreaks .............................................................. 6
netherlands ............................................................... 7
openmap ................................................................. 9
scaleBar ................................................................. 11

Index 14
colourScale

Create colour scales

Description

Produces a scale of colours for plotting maps

Usage

colourScale(x, breaks=5, style=c("quantile","equal","unique", "fixed"),
col="YlOrRd", opacity=1, dec=NULL, firstBreak=NULL,
transform=NULL, revCol=FALSE, exclude=NULL, labels=NULL, ...)

Arguments

x A vector or single-layer Raster, numeric or factor, for which a colour scale will be created
breaks For `colourScale` either the number of or vector of breaks. For `legendBreaks` usually the output of `colourScale`, or a vector of breaks
style Style for breaks, see Details
col Colours to use, either a function or argument for `brewer.pal`
opacity adds transparency to colours, either a single number, vector of length 2, or vector of same length as `breaks`
dec Number of decimal places for the breaks
firstBreak If non-null, force the first break to take this value (often zero).
transform A list of two functions to transform `x` and inverse transform the breaks, or a numeric value specifying a Box-Cox parameter.
revCol Reverse the order of the colours.
exclude A vector of values to change to NA when they appear in `x`
labels Vector of names of levels, useful when `style`='unique'
... Additional arguments passed to `classIntervals`.

Details

colourScale produces intervals from `x`, each with a unique colour. Categories are determined with break points according to the following style options:

• quantile: `quantile(x, prob=seq(0,1,len=breaks), )`

equal: `seq(min(x), max(x), len=breaks)`
unique: `sort(table(unique(x)))[1:breaks]`
fixed: `breaks`
any other string: is passed to `classIntervals`
**colourScale**

**Value**

A list with elements

- **plot**: Vector of same length of x containing colours (RGB hex)
- **breaks**: vector of break points
- **col**: vector of unique colour values corresponding to **breaks**
- **colWithOpacity**: as **col**, but with two digit transparency values appended.

**See Also**

`legendBreaks`, `scaleBar`, `classIntervals`

**Examples**

```r
Npoints = 20
myPoints = SpatialPointsDataFrame(20*cbind(runif(Npoints), runif(Npoints)),
  data=data.frame(y1=c(NA, rnorm(Npoints-1)),
  y2=c(sample(1:4, Npoints-1,replace=TRUE), NA)),
  proj4string=CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0")
)

## Not run:
mymap = openmap(myPoints)

## End(Not run)

if(require('RColorBrewer', quietly=TRUE)) {
  theCol = 'RdYlBu'
} else {
  theCol = heat.colors
}

myscale = colourScale(myPoints$y1, breaks=4, col=theCol,
  style="quantile", revCol=TRUE, dec=1)

map.new(myPoints)
## Not run:
plot(mymap,add=TRUE)

## End(Not run)
plot(myPoints, col=myscale$plot, pch=16,add=TRUE)
legendBreaks("topleft", breaks=myscale)

myscale2 = colourScale(myPoints$y1, breaks=8, col=rainbow, style="equal",
  opacity=0.8, dec=2, revCol=TRUE)

map.new(myPoints)
## Not run:
```

GNcities

Retrieve city names and locations

Description

This function uses the geonames package to provide city names and locations from www.geonames.org.

Usage

GNcities(north, east, south, west, lang = "en", maxRows = 10)
GNsearch(...)
Arguments

north  A bounding box or SpatialPoints or SpatialPolygons or Extent or Raster object, or a decimal degree of longitude.

east, south, west  If north is numeric, decimal degree bounding box.

lang  Language for internationalised returned text

maxRows  Limit on returned rows

...  Various search arguments

Value

A SpatialPointsDataFrame with the same projection north if it exists, otherwise in long-lat.

See Also

GNcities, GNsearch

Examples

myraster = raster(matrix(0,10,10),xmin=8,xmax=18,ymin=0,ymax=10, crs="+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0")
values(myraster) = seq(0,1,len=ncell(myraster))
myPoints = SpatialPoints(myraster, proj4string=CRS(proj4string(myraster)))[
  seq(1,ncell(myraster),len=5)]

## Not run:

if (requireNamespace("geonames", quietly = TRUE)) {
  cities=GNcities(myPoints, max=5)
  mytiles = openmap(myraster)

  map.new(cities)
  plot(mytiles, add=TRUE)
  points(cities, col='red')
  text(cities, labels=cities$name, col='red', pos=4)

  cities=GNcities(myraster, max=5)

  map.new(cities)
  plot(mytiles, add=TRUE)
  points(cities, col='red')
  text(cities, labels=cities$name, col='red', pos=4)

  GNsearch(qe="Toronto Ontario")
}

## End(Not run)
LegendBreaks

Legends for colour scale

Description

Legends where N+1 labels are supplied as the limits of N bins.

Usage

```r
legendBreaks(pos, breaks, col=breaks$col,
            legend=breaks$breaks, rev=TRUE, outer=TRUE,
            pch=15, cex=par('cex'), pt.cex=2.5*cex, inset=0.05,
            text.col=par('fg'), title.col=text.col, adj=0,
            ...)
```

Arguments

- **pos** Position, as specified in the `legend` function.
- **breaks** Optional list with elements `col` and `legend`, such as the output from `colourScale`.
- **col** Single colour or vector of colours for each bin.
- **legend** vector of labels for the legend, one more element than there are colours.
- **rev** if TRUE, labels and colours are ordered from bottom to top, otherwise top to bottom.
- **outer** If TRUE, put legend in the margin of the plot.
- **pch** see `legend`.
- **cex** see `legend`.
- **pt.cex** see `legend`.
- **inset** see `legend`.
- **text.col** see `legend`.
- **title.col** see `legend`.
- **adj** Adjustment of the legend labels relative to plotting symbols.
- **...** Additional arguments passed to `legend`.


Details

A legend for 'z-axis' colour scales.

Value

Result of call to \texttt{legend}

See Also

colourScale

\begin{longtable}{ll}
\texttt{netherlands} & \textit{Data from the Netherlands} \\
\end{longtable}

Description

Elevation data and map tiles for the Netherlands

Usage

data("netherlands")

Format

\texttt{nldElev} is a raster of elevation \texttt{nltTiles} is a background map \texttt{nldCities} is a \texttt{SpatialPointsDataFrame} of city locations.

Details

The inclusion of these datasets is intended to allow the package to build when an internet connection is not present.

Source

See examples.

See Also

\texttt{meuse, getData, openmap}
Examples

```r
# soil data
library("sp")
data("meuse")
coordinates(meuse) <- ~x+y

if(require('rgdal', quietly=TRUE)) {
  proj4string(meuse) <- CRS("+init=epsg:28992")
} else {
  proj4string(meuse) <- CRS(  
    paste("+proj=sterea +lat_0=52.15616055555555 +lon_0=5.387638888888889",  
      "+k=0.9999979 +x_0=155000 +y_0=463000 +ellps=bessel +units=m +no_defs")
)
}

meuse$soilFac = factor(meuse$soil, levels=c(1,2,3),
labels=c("Calcareous","Non-Calc's","Red Brick"))
soilCol = colourScale(meuse$soilFac)
data("netherlands")

map.new(meuse)
plot(nldTiles,add=TRUE)
points(nldCities)
text(nldCities,label=nldCities$name, pos=2)
points(meuse, pch=16, col=soilCol$plot)
legend('topleft', fill=soilCol$col, legend=soilCol$legend)
insetMap(meuse, "bottomright",map=world)

# location won't be marked on the inset map unless rgdal is available

## Not run:
# this is how the data were obtained

# map tiles
nldTiles = openmap(meuse, zoom=12)

# cities
nldCities = GNCities(nldTiles, maxRows=25)

# world
world = openmap(extent(-10,30,40,60))

# elevation data
require('rgdal')
meuseLL = spTransform(meuse, CRS("+init=epsg:4326"))
getData("SRTM", lon=xmin(extent(meuseLL)),  
lat=ymin(extent(meuseLL)),path=tempdir())
nldElev = raster(paste(tempdir(), "/", "srtm_38_02.tif", sep=""))
nldElev = crop(nldElev, extend(extent(meuseLL), 0.1))
```
openmap

nldElev = projectRaster(nldElev, crs=proj4string(meuse))
nldElev = crop(nldElev, extent(nldTiles))

# save the files where the package builder wants them
# save(nldElev, nldTiles, nldCities, world,
# file="~/workspace/diseasemapping/pkg/mapmisc/data/netherlands.RData",
# compress="xz")

## End(Not run)

openmap  Download map tiles

Description

Downloads map tiles from Openstreetmap.org and other servers.

Usage

```r
openmap(x, zoom,
path="http://tile.openstreetmap.org/",
maxTiles = 9,
crs=NA, extend=0, verbose=FALSE)
getTiles(xlim, ylim, zoom, path, maxTiles = 16, cacheDir=tempdir(),
timeOut=5*24*60*60, verbose=FALSE)
oSMTiles(name)
```

Arguments

- **x**: Either raster, SPDF, matrix corresponding to a bounding box or an Extent.
- **zoom**: the zoom level, when missing it will be determined by maxTiles.
- **path**: the tile server from which to get the map, see [http://wiki.openstreetmap.org/wiki/Tiles#Servers](http://wiki.openstreetmap.org/wiki/Tiles#Servers).
- **maxTiles**: If zoom is null, zoom will be chosen such that the number of map tiles is less than or equil to this number.
- **crs**: Projection for the output, defaulting to the same projection as x. If x has no projection, for instance when x is a matrix or extent, crs is also used as the projection of x. If crs is missing and x has no crs, long-lat is used.
- **extend**: Extend the extent for which the map is requested, in units of x. Can be negative, or a vector of length 2 for different x and y extensions.
- **cacheDir**: A directory for caching tiles retrieved over the web. Not used if tiles are coming from a local directory.
- **xlim**: Length-2 vector of min and max X coordinate (latitude)
- **ylim**: Length-2 vector of min and max Y coordinate (longitude)
- **timeOut**: Tiles in the cache directory older than this (in minutes) are re-retrieved.
verbose  If TRUE, give information on where tiles are coming from, cache hits, etc
name  name of a tile path, if missing a vector of all available tile paths is returned. name can be any of the names of the vector returned when name is unspecified.

Details

These functions download, display, and manipulate map tiles stored in a standard way either on a web server or a local folder.

Map tiles are a set of PNG images that span the world at a set of zoom levels. Zoom level 0 has one tile that covers the whole world at a resolution of 256x256 pixels. Zoom level 1 has four 256x256 pixel tiles in a 2x2 pattern over the whole world. In general, zoom level n has $2^n$ by $2^n$ tiles. Zoom levels go up to about 17 or 18 depending on the tile server.

The function getTiles() retrieves tiles to cover a given latitude and longitude range at a given zoom level. Note that since tiles are generated at a fixed set of lat-long ranges the returned tiles will overlap the requested area. There may be some way of clipping to the requested area in a future revision.

The getTilePaths() function returns information about tiles in a region at a zoom level. The tile source path is given but tiles are not requested. A list is returned with one element per tile giving the path, the I and J coordinates, the zoom level, and the geographic bounds for the tile.

The tileBbox() function returns the bounds of the tiles by simply constructing the bounding box of all the constituent SpatialGridDataFrame objects.

http://wiki.openstreetmap.org/index.php/Tile_usage_policy

Value

openmap returns a RasterBrick brick, with 'red', 'green' and 'blue' layers. getTiles returns a list of RasterBricks

See Also

openmap

Examples

myraster = raster(matrix(0,10,10),xmn=8,xmx=18,ymn=0,ymx=10,
crs="+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0")
values(myraster) = seq(0,1,len=ncell(myraster))

myPoints = SpatialPoints(myraster, proj4string=CRS(proj4string(myraster)))[
  seq(1,ncell(myraster),len=5)]

names(osmTiles())

## Not run:

mytiles = openmap(myraster)
scaleBar

```r
map.new(myraster)
plot(mytiles, add=TRUE)
points(mypoints, col='red')

mytiles = openmap(mypoints,
        path=osmTiles("mapquest-sat"), verbose=TRUE)
map.new(mypoints)
plotRGB(mytiles, add=TRUE)
points(mypoints, col='red')

## End(Not run)
```

---

**scaleBar**

*Map with scale bar*

---

**Description**

Utilities for plotting a map, adding a scale bar and north arrow, and adding a legend of colour scales.

**Usage**

```r
scaleBar(crs, pos = "bottomright", scale.cex = 1, outer=TRUE,...)
map.new(x,legendRight=FALSE)
insetMap(crs, pos="bottomright",map="osm",zoom=0,
        width=0.2, col="#FF000090", borderSmall=NA, borderBig=NULL,
        cropInset = extent(-170,xmax=170, ymin=-65, ymax=75),
        outer=TRUE)
```

**Arguments**

- `crs` A CRS object, proj4 string, or an object from which a projection can be extracted with `proj4string(crs)`
- `pos` Position, as specified in the `legend` function.
- `scale.cex` Scaling factor for the scale bar.
- `x` A spatial object from which an extent can be extracted.
- `legendRight` Leave room to the right for the legend produced by plotting a Raster object
- `map` Either a Raster for the inset map or a string passed to `openmap`'s path argument
- `zoom` Zoom level if retrieving inset map from `openmap`
- `width` Width of the inset map, as a fraction of the plot window
- `col` Colour for shaded region of inset map
borderSmall, borderBig
    border style for the inset map and shaded region, passed as the bg argument to
    polygon
cropInset  Crop the inset map to this extent
outer  If TRUE, put bar or map in the margin of the plot
...  breaks = c(-100, -1, 1, Inf) thecol = c("red", "orange", "blue")
    Additional arguments passed to legend, for example bty = "n"

Details

scaleBar produces a scale bar reflecting the distance travelling on a great circle from the centre
of the plot and travelling to the right. The length of the bar is the width of 6 characters times
scale.cex.

map.new initiates a plot intended to contain a map covering the extent of x, with no margins.

Author(s)

Patrick Brown

See Also

scalebar, spplot

Examples

Npoints = 20
myPoints = SpatialPointsDataFrame(
    cbind(runif(Npoints), 51+runif(Npoints)),
    data=data.frame(y1=c(NA, rnorm(Npoints-1)),
    y2=c(sample(0:5, Npoints-1,replace=TRUE), NA)),
    proj4string=CRS("+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0")
)
)

## Not run:
mymap = openmap(myPoints)

## End(Not run)

breaks = c(-100, -1, 1, Inf)
thecol = c("red", "orange", "blue")

map.new(myPoints)
## Not run:
plot(mymap, add=TRUE)

## End(Not run)
plot(myPoints, col = as.character(cut(myPoints$y1, breaks, thecol)), add=TRUE)
scaleBar(myPoints, "bottomright", cex=1.25, scale.cex=2)
temp=legendBreaks("topleft", legend=breaks, col=thecol)

## Not run:
insetMap(myPoints, "bottomleft", col='blue')

## End(Not run)
Index

*Topic datasets
  netherlands, 7

brewer.pal, 2
brick, 10

classIntervals, 2, 3
colorScale (colourScale), 2
colourScale, 2, 6, 7

data, 7
tiles (openmap), 9
GNcities, 4, 5
GNsearch, 5
GNsearch (GNcities), 4

insetMap (scaleBar), 11

legend, 6, 7
legendBreaks, 3, 6

map.new (scaleBar), 11
meuse, 7

netherlands, 7
nldCities (netherlands), 7
nldElev (netherlands), 7
nldTiles (netherlands), 7

openmap, 7, 9, 10, 11
osmTiles (openmap), 9

polygon, 12

scaleBar, 3, 11
scalebar, 12
spplot, 12

world (netherlands), 7