Package ‘mapmisc’

May 21, 2015

Type Package
Title Utilities for Producing Maps
Version 1.2.4
Date 2015-05-21
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
NeedsCompilation no
Repository CRAN
Date/Publication 2015-05-21 20:30:47

R topics documented:

col2html .......................................................... 2
colourScale .................................................... 3
GNcities ........................................................ 5
legendBreaks .................................................... 7
legendTable ..................................................... 8
netherlands ..................................................... 9
omerc ............................................................. 11
openmap ........................................................ 13
scaleBar ........................................................ 14
tpeqd ............................................................ 16

Index 18
col2html

Convert colours to HTML hex

Description

Converts any object interpretable as a colour to an HTML hex string, i.e. ‘red’ to ‘#FF0000’.

Usage

col2html(col, opacity=1, alpha)

Arguments

col Either a character vector of colour names as listed by \texttt{colours()} or an integer vector of colour indexes. Passed to \texttt{col2rgb}.

opacity scalar or vector of colour opacities between 0 and 1.

alpha Integer between 0 and 255, or a character giving a 2-digit hex value. Overrides opacity and passed to \texttt{rgb}.

Value

A vector of 6 or 8 digit hex codes specifying HTML colours.

See Also

\texttt{col2rgb.rgbhexmode}

Examples

\begin{verbatim}
col2html(1:10)
col2html(c(‘red’, ‘blue’), 0.5)
col2html(c(2, 4), 0.5)
col2html(c(stuff=’red’, foo=’blue’), alpha=128)
col2html(c(‘red’, ‘blue’), alpha=’80’)  
col2html(c(2, 4), alpha=’80’)
N = length(palette())
plot(1:N, rep(1,N), xlim=c(0,N), pch=16, cex=5, 
col=col2html(1:N))
points(1:N, rep(1,N), pch=15, cex=4.5, col=palette())
text(-0.5+1:10, rep(1,10), col2html(1:10), srt=90)
text(1:N, rep(0.7,N), palette())
text(1:N-0.5, rep(1.3, N), col2html(palette()), cex=0.7)
\end{verbatim}
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 brewerPal
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
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

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

### GNcities

**Retrieve city names and locations**

**Description**

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

**Usage**

```r
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

```r
myraster = raster(matrix(0,10,10),xmin=8,xmax=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)]

## 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)

  mapmisc::GNsearch(q="Toronto Ontario")
}

## End(Not run)
```
**legendBreaks**

---

**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, legend,  
rev=TRUE,  
outer=TRUE,  
pch=15,  
bg=par('bg'),  
cex=par('cex'),  
pt.cex=2.5*cex,  
text.col=par('fg'),  
title=NULL,  
inset=0.05,  
title.col=text.col,  
adj=0,  
y.intersp,  
...)
```

**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`.

- `bg`  
  Background colour see `legend`.

- `cex`  
  See `legend`.

- `pt.cex`  
  See `legend`.

- `text.col`  
  See `legend`.

- `title`  
  See `legend`.

- `inset`  
  See `legend`.
title.col  see `legend`
adj  Adjustment of the legend labels relative to plotting symbols.
y.interp  see `legend`
...  Additional arguments passed to `legend`.

Details
A legend for 'z-axis' colour scales.

Value
Result of call to `legend`  

See Also
`colourscale`

data.frame or character vector

legendTable  `Table for colour scales`

data.frame or character vector

Description
A table in html or Latex showing values associated with colours

Usage

legendTable(x,  
    type=c('latex', 'html'),  
    box = c(-0.2, 1, 2),  
    unit = 'em',  
    collapse=NULL)

Arguments

x  a `data.frame` with columns `col` and `label`, possibly produced by `colourscale`
type  html or latex compatible output
box  dimensions of colour boxes, passed as depth, height and width to `rule` in Latex, or width (first two elements ignored) for html.
unit  Units for box dimensions
collapse  If non-NULL, passed to `paste` to produce a character vector instead of table

Value

netherlands

See Also

colourScale

Examples

mytable = data.frame(col=col2html(1:5), label=1:5)

legendTable(mytable)
legendTable(mytable, collapse=': ')
legendTable(mytable, type='html')

netherlands  Data from the Netherlands

Description

Elevation data and map tiles for the Netherlands

Usage

data("netherlands")

Format

nldelev is a raster of elevation nlttiles is a background map nlcdcities is a 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

time {datasets}, 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=stere +lat_0=52.156160555555555 +lon_0=5.38763888888889",
    "+k=0.9999079 +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=min(extent(meuseLL)),
lat=min(extent(meuseLL)),path=tempdir())
nldElev = raster(paste(tempdir(), "/", "srtm_38_02.tif", sep=""))
nldElev = crop(nldElev, extend(extent(meuseLL), 0.1))
```
omerc

## Oblique mercator projections

### Description

Defines an appropriate oblique mercator projection for a spatial object

### Usage

```r
omerc(x, angle,
    post=c('none', 'north', 'wide', 'tall'),
    preserve=NULL)
```

### Arguments

- `x` A SpatialP* object or a vector of length 2 giving the centroid of the projection.
- `angle` angle of rotation or vector of angles
- `post` post-projection angle rotation
- `preserve` A SpatialPoints object, the resulting projection is scaled so as to preserve the distances between these points as best as possible.

### Details

A map projection is produced which warps the world onto a cylinder, with the north-south axis rotated by the specified angle. If angle is a vector, the optimal angle for reducing the size of the bounding box is returned.

If `post = 'north'`, an inverse rotation will preserve the north direction at the origin.

If `post = 'wide'`, an inverse rotation makes the smallest possible bounding box which is wider than tall.

If `post = 'tall'`, the bounding box is taller than it is wide.

If `post` is numeric, it specifies an angle for inverse rotation.

### Value

An object of class `CRS`.

```r
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)
```
References


See Also

CRS, spTransform

Examples

omerc(c(10, 50), angle=c(0, 45, 80))

data('netherlands')

if(require('rgdal', quietly=TRUE)){
  nldUtm = spTransform(nldCities, omerc(nldCities, angle=0))
  projection(nldUtm)

  map.new(nldUtm)
  text(nldUtm, labels=nldUtm$name)
  scaleBar(nldUtm, 'topright')

  nldRot = spTransform(nldCities, omerc(nldCities, angle=seq(25, 45, by=1))
  )
  projection(nldRot)

  map.new(nldRot)
  text(nldRot, labels=nldRot$name)
  scaleBar(nldRot, 'topright')
  insetMap(nldRot, 'bottomright', map=world)
}

## Not run:

map.new(nldCities)
plot(nldTiles, add=TRUE)

# Run:

omerc(c(10, 50), angle=c(0, 45, 80))

data('netherlands')

if(require('rgdal', quietly=TRUE)){
  nldUtm = spTransform(nldCities, omerc(nldCities, angle=0))
  projection(nldUtm)

  map.new(nldUtm)
  text(nldUtm, labels=nldUtm$name)
  scaleBar(nldUtm, 'topright')

  nldRot = spTransform(nldCities, omerc(nldCities, angle=seq(25, 45, by=1))
  )
  projection(nldRot)

  map.new(nldRot)
  text(nldRot, labels=nldRot$name)
  scaleBar(nldRot, 'topright')
  insetMap(nldRot, 'bottomright', map=world)
}

## Not run:

map.new(nldCities)
plot(nldTiles, add=TRUE)

# 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=projection(x), extend=0, verbose=FALSE) 
openmapTiles(name) 
openmapAttribution(name, type=c('text','latex','markdown','html'), short=FALSE)
```

**Arguments**

- **x**: An **extent** or any spatial object (raster, Spatial*) from which an extent can be obtained.
- **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 missing, `zoom` will be chosen such that the number of map tiles is less than or equal 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.
- **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.
- **type**: format for the attribution
- **short**: short or long attribution

**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 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.

Be sure to attribute any maps you publish, the `osmAttribution` function will assist.
Value

openmap returns a RasterBrick brick, with 'red', 'green' and 'blue' layers.

See Also

openmap

Examples

```r
myraster = raster(matrix(c(0,1,10),xmn=8,xmx=18,ymn=8,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)

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')
openmapAttribution(mytiles)

## End(Not run)

openmapAttribution("osm", type='markdown')
```

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.
scaleBar

Usage

scaleBar(crs, pos = "bottomright",
        cex=par("cex"),
        pt.cex = 1.25*cex,
        seg.len=8*cex,
        title.cex=cex,
        outer=TRUE,...)
map.new(x,legendRight=FALSE)
insetMap(crs, pos="bottomright",map="osm",zoom=0,
        width=max(c(0.2, 1-par("plt")[2])),
        col="#FF000090", borderSmall=NA, borderBig=NA,
        cropInset = extent(-180, xmax=180, ymin=-47, ymax=71),
        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.
cex scaling factor for the legend
pt.cex Scaling factor north arrow (can be zero).
seg.len approximate length (in character units) of the scale bar. can be zero.
title.cex scaling for the distance text
outer If TRUE, put bar or map in the margin of the plot
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
... 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, splot

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)

---

tpeqd

Two point equidistant projections
**tpeqd**

**Description**
Defines map projection

**Usage**
tpeqd(x, offset=c(0,0))

**Arguments**
x A SpatialPoints* object of length 2 or a matrix with two columns.
offset 2 coordinates to define the origin

**Details**
A coordinate reference system is returned

**Value**
An object of class CRS.

**References**
http://en.wikipedia.org/wiki/Two-point_equidistant_projection

**See Also**
CRS, spTransform

**Examples**

tpeqd(rbind(c(0,0), c(10,50)))
data('netherlands')
tcrs = tpeqd(nldCities[1:2,])
tcrs

if(require('rgdal', quietly=TRUE)) {
  nldT = spTransform(nldCities, tcrs)
  projection(nldT)
  map.new(nldT)
text(nldT, labels=nldT$name)
scaleBar(nldT, 'topright')
}
Index

*Topic datasets
   netherlands, 9

brewer.pal, 3
brick, 14
classIntervals, 3, 4
col2html, 2
col2rgb, 2
colorScale (colourScale), 3
colours, 2
colourScale, 3, 7–9
CRS, 11, 12, 17

extent, 13

getData, 9
GNCities, 5, 6
GNsearch, 6
GNsearch (GNCities), 5

hexmode, 2

insetMap (scaleBar), 14

legend, 7, 8
legendBreaks, 4, 7
legendTable, 8

map.new (scaleBar), 14
meuse, 9

netherlands, 9
nldCities (netherlands), 9
nldElev (netherlands), 9
nldTiles (netherlands), 9

omerc, 11
openmap, 9, 13, 14, 15
openmapAttribution (openmap), 13
osmTiles (openmap), 13

polygon, 15
rgb, 2
scaleBar, 4, 14
scalebar, 16
spplot, 16
spTransform, 12, 17
tpeqd, 16
world (netherlands), 9