Package 'mapmisc'

April 16, 2024

Type Package

Title Utilities for Producing Maps

Version 2.1.0

Date 2024-04-15

Depends terra, R (>= 3.5.0)

Imports methods, grDevices, stats, utils, graphics, geosphere

Suggests RColorBrewer, geonames, classInt, knitr

Enhances XML, RCurl

Description

Provides a minimal, light-weight set of tools for producing nice looking maps in R, with support for map projections. See Brown (2016) <doi:10.32614/RJ-2016-005>.

License GPL

LazyData true

LazyDataCompression xz

VignetteBuilder knitr

NeedsCompilation no

Author Patrick Brown [aut, cre, cph]

Maintainer Patrick Brown <patrick.brown@utoronto.ca>

Repository CRAN

Date/Publication 2024-04-16 06:20:03 UTC

R topics documented:

`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 `colours()` or an integer vector of colour indexes. Passed to `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 `rgb`.

Value

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

See Also

`col2rgb`,`rgbhexmode`

Examples

```
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)
```

`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, digits = 6, firstBreak=NULL,
 transform=NULL, revCol=FALSE, exclude=NULL, labels=NULL, ...)
colorScale(...)
breaksForRates(x, breaks = 10, transform = 0.1,
      multiples = c(2, 4, 5, 10))
```

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
`digits`	Number of significant figures
`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'
multiples	break points must be multiples of these numbers times a power of 10
...	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

colorScale passes all it's arguments to colourScale

breaksForRates returns break points suitable for mapping incidence rates, which are positive and always include 1.0.

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

```
breaksForRates(13.6, breaks = 7)

Npoints = 20

myPoints = vect(
cbind(runif(Npoints), 51+runif(Npoints)),
atts=data.frame(
y1=c(NA, rnorm(Npoints-1)),
```

```
    y2=c(sample(0:5, Npoints-1,replace=TRUE), NA)
),
crs=crsLL)



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

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


data("netherlands")
nldElev = terra::unwrap(nldElev)
myscale = colourScale(nldElev, breaks=4, col=theCol, style='equal', dec=0)

oldpar = map.new(myPoints)
plot(myPoints, col=myscale$plot, pch=16,add=TRUE)
legendBreaks("topleft", myscale)


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

map.new(myPoints)
plot(myPoints, col=myscale2$plot, pch=16,add=TRUE)
legendBreaks("topleft", myscale2)

if(require('RColorBrewer', quietly=TRUE)) {
theCol = 'Set2'
} else {
theCol = grDevices::heat.colors
}

myscale3 = colourScale(myPoints$y2,  breaks=3,col=theCol, style="unique",
opacity=c(0.1, 0.9))

map.new(myPoints)
plot(myPoints, col=myscale3$plot, pch=16,add=TRUE)
legendBreaks("topleft", myscale3)

myPoints$y3 = exp(myPoints$y1)
myscale4 = colourScale(myPoints$y3,  breaks=4, style="equal",
opacity=c(0.1, 0.9), transform=1.25,dec=0, firstBreak=0)

map.new(myPoints)
plot(myPoints, col=myscale4$plot, pch=16,add=TRUE)
legendBreaks("topleft", myscale4)
```

```
# raster with colour table

x = rast(extent=ext(0,15,0,10), res=1)
values(x) = sample(1:4, ncell(x), replace=TRUE)
myScale = colourScale(x, breaks=3, style='unique', col=c('red','blue','orange'))
if(utils::packageVersion("terra") >= "1.7-40" ) {
terra::coltab(x) = myScale$colourtable
plot(x)
} else {
plot(x, breaks = myScale$breaks, col=myScale$col)
}
legendBreaks('topright', myScale)

par(oldpar)
```

`crsMerc` *Some coordinate reference systems and bounding boxes*

Description

Defines CRS's for the several map projections.

Usage

```
crsMerc
crsLL
crsCanada
extentMerc
bboxLLsafe
bboxLL
```

Format

`crsMerc` spherical Mercator projection used by web mapping services, epsg:3857 `crsLL` long-lat, epsg:4326 `crsCanada` customized oblique mercator for Canada `bboxLL` polygon of bounding box of long-lat, -180 to 180, -90 to 90 `bboxLLsafe` as bboxLL, but slightly away from the edges `extentMerc` extent of spherical mercator projections

Details

these objects are used internally and may be of interest to the user

Value

objects of class `crs` or numeric vectors.

References

https://en.wikipedia.org/wiki/Web_Mercator, https://spatialreference.org/ref/epsg/4326/

See Also

crs

Examples

```
terra::crs(crsMerc, proj=TRUE)
terra::crs(crsLL, proj=TRUE)
terra::crs(crsCanada, proj=TRUE)
terra::ext(extentMerc)

bboxLLsafe = terra::unwrap(bboxLLsafe)
plot(bboxLLsafe)
plot(terra::project(bboxLLsafe, crsMerc))
```

---

geocode *Georeferencing with Google*

Description

Uses the dismo package to geocode with Google

Usage

```
geocode(x, extent,
  lang = gsub("(_|[:]).*", "", Sys.getenv('LANGUAGE')))
```

Arguments

x	Vector of character strings to search for
extent	Currently unused. an Extent object, or any object from which an Extent can be obtained.
lang	Language for place names in result.

Details

If the option `getOption('mapmiscCachePath')` is set, it will be used to specify the folder to save downloaded data. `getOption('mapmiscVerbose')` for printing progress.

Data are retreived from Openstreetmap.org, see https://wiki.openstreetmap.org/wiki/Nominatim.

Value

A SpatialPointsDataFrame with coordinates in the projection of extent if possible, or long-lat otherwise.

Examples

```
cities=try(geocode('Ulan batar'), silent=TRUE)
data('worldMap')
worldMap = terra::unwrap(worldMap)

if(!all(class(cities) == 'try-error') ) {
citiesT = project(cities, crs(worldMap))
oldpar=map.new(citiesT, buffer=5000*1000)
plot(worldMap, add=TRUE)
points(citiesT, col='red')
suppressWarnings(text(citiesT, labels=citiesT$name, col='red',pos=4))
## Not run:
# uses unicode symbols
text(citiesT, labels=citiesT$display_name, col='red',pos=1))

## End(Not run)
par(oldpar)
}
```

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, buffer=0)
GNsearch(..., crs=crsLL)
```

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
buffer	passed to extend
...	Various search arguments
crs	projection for the output

Value

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

See Also

GNcities, GNsearch

Examples

```
## Not run:
GNsearch(q="Toronto Ontario", maxRows = 3)

## End(Not run)


library('terra')
myraster = rast(
matrix(1:100,10,10),
extent=ext(8,18,0,10), crs=crsLL)

options(geonamesUsername="myusernamehere")
if(file.exists("~/geonamesUsername.R")) source("~/geonamesUsername.R")

if(requireNamespace("geonames", quietly = TRUE)) {

cities=try(GNcities(myraster, max=5), silent=TRUE)
mytiles = openmap(myraster, zoom=5, buffer=1)

oldpar=map.new(mytiles)
plot(mytiles, add=TRUE)
if(!all(class(cities)=='try-error')) {
points(cities, col='red')
text(cities, labels=cities$name, col='red',pos=4)
}


par(oldpar)

}
```

gridlinesWrap *Adds long-lat grid for projected data*

Description

long-lat grid lines are added to a map in the coordinate system specified, allowing for map projections wrapped differently from the 180 meridian.

Usage

```
gridlinesWrap(crs,
easts=seq(-180,180,by=60),
norths=seq(-90,90,by=30),
ndiscr=40, plotLines=TRUE,
plotLabels = TRUE, ...)
```

Arguments

crs	A character string representing a CRS
easts	vector of longitudes
norths	vector of latitudes
ndiscr	number of intermediate points per line
plotLines	add lines to existing plot
plotLabels	add labels to existing plot
...	Additional arguments passed to lines or text, for example lty=2

Value

A list with elements lines, containg the graticule lines, and points containing the locations and labels for longitude and latitude values.

Author(s)

Patrick Brown

See Also

graticule

Examples

```
data('worldMap')
worldMap = terra::unwrap(worldMap)
crsMoll = moll(-100)
worldMapT = wrapPoly(worldMap, crsMoll, buffer.width=200*1000)
```

```
plot(attributes(crsMoll)$ellipse)
plot(worldMapT, add=TRUE)
gridlinesWrap(crsMoll, lty=3, col='red', cex=0.6)
```

`legendBreaks` *Legends for colour scale*

Description

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

Usage

```
legendBreaks(pos,
breaks,
col,    legend,
    rev=TRUE,
    outer=TRUE,
    pch=15,
    bg='white',
    cex=par('cex'),
    pt.cex=2.5*cex,
    text.col=par('fg'),
    title=NULL,
    inset=0.05,
    title.col=text.col,
    adj=0,
    width=Inf,
    lines=Inf,
    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.
width	Maximum number of characters before a line break is added to the legend labels
lines	Maximum number of lines in each legend label
y.intersp	see legend
...	Additional arguments passed to legend.

Details

A legend for 'z-axis' colour scales.

Value

Result of call to legend

See Also

colourScale

---

legendTable *Table for colour scales*

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

data.frame or character vector

See Also

colourScale

Examples

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

legendTable(mytable)
legendTable(mytable, collapse=';')
legendTable(mytable, type='html')
```

---

map.new *Start a new map*

Description

Prepare a plotting window suitable for a map

Usage

```
map.new(x,legendRight=FALSE, buffer=0, mar=c(0,0,0,0), ...)
```

Arguments

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
buffer	passed to extend to increase the plotting area
mar	see par
...	Additional arguments passed to plot

Details

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

Value

A list of the graphical parameters prior to calling map.new

Author(s)

Patrick Brown

Examples

```
nldTiles = terra::unwrap(nldTiles)
nldCities = terra::unwrap(nldCities)


oldpar = map.new(nldCities)
plot(nldTiles, add=TRUE)
points(nldCities)
par(oldpar)
```

---

modis *MODIS tiles and projection*

Description

Raster containing MODIS tile ID's

Usage

```
getModisTiles(x, tiles)
crsModis
getModisRaster()
getDegreeRaster()
```

Arguments

x	A spatial object which modis tiles will cover.
tiles	A raster with modis (or other) tiles.

Details

Provides information on tiles which can be downloaded from MODIS.

Value

`getModisTiles` returns a matrix with modis tiles.

`getModisRaster` shows horizontal and vertical tile names for downloading data from MODIS.

`getDegreeRaster` shows horizontal and vertical tiles in long-lat, for downloading elevation.

References

https://modis-land.gsfc.nasa.gov/MODLAND_grid.html, https://spatialreference.org/ref/esri/54008/

Examples

```
crsModis

myPointLL = vect(cbind(c(5:6),10:11), crs = crsLL)


getModisTiles(myPointLL)

getModisTiles(myPointLL, getDegreeRaster())



modisUrl =  'https://e4ftl01.cr.usgs.gov/MOTA/MCD12Q1.061/2002.01.01/'
desiredTiles = paste0("(",
paste(getModisTiles(myPointLL, getModisRaster())[,'tile'], collapse='|'),
").*.hdf$")

if(requireNamespace("RCurl", quietly=TRUE) & requireNamespace("XML", quietly=TRUE)) {
allFiles = try(XML::getHTMLLinks(RCurl::getURL(
    modisUrl,ftp.use.epsv=FALSE,
    dirlistonly = TRUE)), silent=TRUE)
if(!identical(class(allFiles), 'try-error')) {
theFiles =  grep(desiredTiles, allFiles, value=TRUE)
paste0(modisUrl, theFiles)
}
}
```

---

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 `meuse` classic Meuse river data set from the sp package `nldCities` 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.

Examples

```
meuse = terra::unwrap(meuse)
nldTiles = terra::unwrap(nldTiles)
nldCities = terra::unwrap(nldCities)

oldpar=map.new(meuse, buffer=1*1000)
plot(nldTiles,add=TRUE)
points(nldCities, pch=4, col='blue')
text(nldCities,label=nldCities$name, pos=2, col='blue')
points(meuse, pch=15, col=as.integer(meuse$soil))

legend('topleft', fill=1:nlevels(meuse$soil),
legend=levels(meuse$soil), inset=0.2, bg='white', title='Soil type')
par(oldpar)
```

---

omerc *Oblique Mercator, Cylindrical, and Mollweide projections*

Description

Defines an appropriate Oblique Mercator, Oblique Cylindrical Equal Area, and Mollweide projections for a supplied Spatial object

Usage

```
omerc(x, angle,
post=c('none', 'north', 'wide','tall'),
    preserve=NULL, ellipse=TRUE)
    ocea(x, angle, flip=FALSE)
    moll(x=0, angle=NULL, flip=FALSE)
```

Arguments

x	A SpatVector 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
flip	post-projection flipping of coordinates
preserve	A SpatVector object, the resulting projection is scaled so as to preserve the distances between these points as best as possible.
ellipse	compute projection region and areas to crop when projecting.

Details

With omerc, an Oblique Mercator 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.

ocea produces an Oblique Cylindrical Equal Area projection and moll a Mollweide projections

Value

An object of class crs.

References

https://en.wikipedia.org/w/index.php?title=Space-oblique_Mercator_projection

See Also

crs, project

Examples

```
data('worldMap')
worldMap = terra::unwrap(worldMap)


myProj = omerc(c(-100,-70), angle=-45)
crs(myProj, proj=TRUE)

plot(project(worldMap, crsLL))
plot(attributes(myProj)$crop, col='red', add=TRUE)
```

openmap *Download map tiles*

Description

Downloads map tiles from Openstreetmap.org and other servers.

Usage

```
openmap(x, zoom,
  path="http://tile.openstreetmap.org/",
  maxTiles = 9,
  crs=ifelse(is.numeric(x), mapmisc::crsLL, terra::crs(x)),
  buffer=0, fact=1,
verbose=getOption('mapmiscVerbose'),
  cachePath=getOption('mapmiscCachePath'),
  suffix=NULL
)

osmTiles(name, xyz, suffix)

openmapAttribution(name,
  type=c('text','latex','markdown','html', 'auto'),
  short=FALSE)
```

Arguments

x	An a spatial object from which an extent and crs can be obtained.
zoom	the zoom level, when missing it will be determined by maxTiles.
path	Source of map tiles, see http://diseasemapping.r-forge.r-project.org/mapLayers.html.
maxTiles	If zoom is missing, zoom will be chosen such that the number of map tiles is less than or equl 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 NA or missing and x has no crs, long-lat is used.
buffer	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
fact	Passed to increase or decrease resolution, values above 1 help to produce a clearer image.
verbose	Print information about map images being downloaded, defaults to FALSE.
cachePath	Location to store downloaded map images, defaults to tempdir()

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
xyz	format of xyz coordinates in URL's
suffix	string to append to URL's, i.e. '.png'

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.

See https://mc.bbbike.org/mc/ for a more possible map tiles (not all of which are compatible with openmap)

Be sure to attribute any maps you publish, the osmAttribution function will assist. If type = 'auto' then markdown format will be used unless a variable mdToTex is defined and equal to TRUE.

Value

openmap returns a SpatRaster with indexed colours or RGB layers.

openmapAttribution returns a character string.

Examples

```
data("netherlands")
nldTiles = terra::unwrap(nldTiles)
plot(nldTiles)

openmapAttribution('osm', short=TRUE, type='markdown')

openmapAttribution("stamen-watercolor", type='text')


myraster = rast(matrix(1:100,10,10),extent=ext(8, 18, 0, 10), crs=crsLL)

myPoints = as.points(myraster)[seq(1, ncell(myraster), len=12)]

names(osmTiles())


mytiles = try(openmap(myraster, zoom=5, verbose=TRUE))

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

```
myPoints = project(myPoints, crsMerc)
map.new(myPoints)

mytiles = try(openmap(myPoints,
path='https://livemap-tiles1.waze.com/tiles', verbose=TRUE, buffer=5))
plot(mytiles, add=TRUE)

points(myPoints, col='red')

par(oldpar)
```

`persistentCache` *Set a persistent cache*

Description

Sets a cache folder in temporary space

Usage

```
persistentCache(verbose=TRUE)
```

Arguments

`verbose`	print location of the cache folder

Details

The default cache for map images is tempdir()/mapmiscCache, which will be deleted when an R session ends. Running this function sets a cache in /tmp/mapmiscCache_[username], which will re-use cached data across R sessions.

Value

`persistentCache` returns the path to the cach folder.

Examples

```
# current cache
getOption("mapmiscCachePath")

# set a new cache
myCache = file.path(tempdir(), 'myCache')
dir.create(myCache)
options(mapmiscCachePath = myCache)
getOption("mapmiscCachePath")
```

```
# create a persistent cache

persistentCache(verbose=TRUE)

getOption("mapmiscCachePath")
```

---

scaleBar *Scale bar and inset map*

Description

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

Usage

```
scaleBar(crs, pos = "bottomright",
cex=1,
    pt.cex = 1.1*cex,
    seg.len=5*cex,
    title.cex=cex,
    outer=TRUE,...)
insetMap(crs, pos="bottomright",map="osm",zoom=0,
width=max(c(0.2, 1-par('plt')[2])),
col="#FF000090", borderMap=NULL,
cropInset = terra::ext(-180, 180,  -47, 71),
outer=TRUE, inset = c(0.1, 0.1), ...)
```

Arguments

crs	A character string from which a projection can be extracted with `terra::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
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
borderMap	border style for the inset map (passed to `polygon`)
cropInset	Crop the insert map to this extent
inset	how far from the border to put the inset map
...	Additional arguments passed to `legend` for `scaleBar` or `polygon` (for `insetMap`).

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.

Value

A list containig coordinates of the elements of the scale bar.

Author(s)

Patrick Brown

Examples

```
Npoints = 20
set.seed(0)
myPoints = vect(
cbind(runif(Npoints)-0.1, 51+runif(Npoints)),
atts=data.frame(
y1=c(NA, rnorm(Npoints-1)),
    y2=c(sample(0:5, Npoints-1,replace=TRUE), NA)
),
crs=crsLL)



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


oldpar = map.new(myPoints)
plot(myPoints,col = as.character(cut(
myPoints$y1, breaks, thecol
)),add=TRUE)
scaleBar(myPoints, "bottomright",cex=1.25, seg.len=2)
legendBreaks("topleft", legend=breaks, col=thecol)

thedot = insetMap(crs=myPoints,
pos="bottomleft",
 col='#00000000', lty=0, outer=FALSE, width=0.25)
points(thedot)

par(oldpar)
```

tonerToTrans *Convert RGB maps to semi-transparent*

Description

Stamen-toner maps are 3-layer RGB rasters, which are converted to single-layer rasters with indexed colours with whites becoming transparent.

Usage

```
tonerToTrans(x,  pattern="(red|green|blue)$", power = 0.5,
col='black',  threshold=Inf,  mostCommon=1)
```

Arguments

x	A RasterStack with RGB colours, such as from openmap with path='stamen-toner'
pattern	string passed to grep to find RGB layers.
power	Values below 1 increase opacity, above 1 increases transparency
col	colour for resulting map
threshold	colours above this value are transparent
mostCommon	integer vector, the most common colours are converted to transparent

Value

A SpatRast with indexed colours

Author(s)

Patrick Brown

See Also

openmap

Examples

```
origMap = openmap(
   c(-11, 9),
  path='cartodb-nolabels',
buffer=2, verbose=TRUE
)


oldpar= map.new(origMap, bg='green')
plot(origMap, add=TRUE)
```

```
transMap = tonerToTrans(origMap, mostCommon=1)
names(transMap)
  map.new(transMap, bg='green')
plot(transMap, add=TRUE)


par(oldpar)
```

---

tpeqd *Two point equidistant and tilted perspective projections*

Description

Defines map projection

Usage

```
tpeqd(x, offset=c(0,0), axis='enu')
tpers(x,  hKm = 100*1000, tilt = -10,   azi, offset=c(0,0), axis='enu')
```

Arguments

x	A SpatialPoints* object of length 2 or a matrix with two columns.
hKm	Height veiwing the Earth from
tilt	Viewing angle
azi	Azimuth, defaults to direction of first two points in x
offset	2 coordinates to define the origin
axis	defaults to east, north, up. 'swu' would rotateo 90 degrees

Details

A coordinate reference system is returned

Value

Caracther string representing a crs.

References

https://en.wikipedia.org/wiki/Two-point_equidistant_projection https://proj.org/operations/projections/tpers.html

See Also

crs,project

Examples

```
data('worldMap');worldMap=unwrap(worldMap)

thepoints = vect(rbind(cbind(150, -40), cbind(-70,-40)), crs=crsLL)
crsOne = tpeqd(thepoints)
worldMapTrans = wrapPoly(worldMap, crsOne)

oldpar=map.new(crsOne, col='lightblue')
plot(worldMapTrans, add=TRUE, col='grey')
points(project(thepoints, crsOne), col='red')
gridlinesWrap(crsOne, col='orange')



thepoints = vect(rbind(cbind(-40, 65), cbind(139,35)), crs=crsLL)
crsTwo = tpeqd(thepoints)


map.new(crsTwo, col='lightblue')
plot(wrapPoly(worldMap, crsTwo), add=TRUE, col='grey')
points(project(thepoints, crsTwo), col='red')
gridlinesWrap(crsTwo, col='orange')

par(oldpar)
```

---

worldMap *Country boundaries*

Description

Country borders from naturalearthdata.com

Usage

```
data("worldMap")
```

Source

https://www.naturalearthdata.com/downloads/110m-cultural-vectors/

Examples

```
# soil data

data("worldMap")
worldMap = terra::unwrap(worldMap)
oldpar=map.new(worldMap)
plot(worldMap, border='red', lwd=3, add=TRUE)
plot(worldMap[worldMap$NAME ==  'Brazil',],
add=TRUE, col='green')
par(oldpar)
```

`wrapPoly` *Reproject polygons with wrapping*

Description

Reprojects a SpatialPolygons object to a projection with longitude wrapping other than 180 degreess

Usage

```
wrapPoly(x,crs, buffer.width = 100*1000)
llCropBox(crs,
  buffer.width=50*1000, densify.interval = 25*1000,
  crop.distance = 2.1e7, crop.poles = FALSE, crop.leftright=FALSE,
  remove.holes=TRUE, cycles = 2, ellipse=NULL)
```

Arguments

`x`	A Spatial object
`crs`	Caracther string representing a `crs`.
`buffer.width`	buffer to add to points on border when cropping poloygons, defaults to 100km
`densify.interval`	interval when densifying
`crop.distance`	crop coordinates larger than this value
`crop.poles`	remove areas near the poles
`crop.leftright`	remove points near 180 longitute line
`remove.holes`	fill holes in the crop region
`cycles`	iterations adding denser points
`ellipse`	boundary of the world in crs coordinates

Value

A reprojected Spatial object.

See Also

project, examples in tpeqd

Index