

# Package ‘mapmisc’

March 11, 2021

**Type** Package

**Title** Utilities for Producing Maps

**Version** 1.8.0

**Date** 2021-03-09

**Depends** sp, raster, R (>= 3.5.0)

**Imports** methods, grDevices, stats, utils, graphics

**Suggests** RColorBrewer, classInt, rgdal, rgeos, geosphere, knitr,  
markdown

**Enhances** tools, geonames

**Author** Patrick Brown <patrick.brown@utoronto.ca>

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

**Description** A minimal, light-weight set of tools for producing nice looking maps in R, with support for map projections.

**License** GPL

**VignetteBuilder** knitr

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2021-03-10 23:50:02 UTC

## R topics documented:

col2html . . . . .	2
colourScale . . . . .	3
crsMerc . . . . .	6
geocode . . . . .	7
GNcities . . . . .	8
gridlinesWrap . . . . .	9
legendBreaks . . . . .	10
legendTable . . . . .	12
map.new . . . . .	13
modis . . . . .	14

netherlands . . . . .	15
omerc . . . . .	16
openmap . . . . .	19
persistentCache . . . . .	21
scaleBar . . . . .	22
tonerToTrans . . . . .	24
tpeqd . . . . .	25
worldMap . . . . .	26
wrapPoly . . . . .	27

<b>Index</b>	<b>28</b>
--------------	-----------

---

col2html	<i>Convert colours to HTML hex</i>
----------	------------------------------------

---

## 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 <a href="#">colours()</a> or an integer vector of colour indexes. Passed to <a href="#">col2rgb</a> .
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 <a href="#">rgb</a> .

## 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 <a href="#">brewer.pal</a>
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 <a href="#">classIntervals</a> .

**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 = 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=mapmisc::crsLL)
```

```
## 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:
plot(mymap,add=TRUE)

## End(Not run)
plot(myPoints, col=myscale2$plot, pch=16,add=TRUE)
legendBreaks("topleft", breaks=myscale2)

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

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

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

## End(Not run)
plot(myPoints, col=myscale3$plot, pch=16,add=TRUE)
legendBreaks("topleft", breaks=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)
## Not run:
plot(mymap,add=TRUE)

## End(Not run)
plot(myPoints, col=myscale4$plot, pch=16,add=TRUE)
legendBreaks("topleft", legend=myscale4$breaks, col=myscale4$col)
```

---

`crsMerc`*Spherical Mercator projection*

---

## Description

Defines CRS's for the Spherical Mercator and long-lat projections.

## Usage

```
crsMerc
crsLL
```

## Details

CRS objects for epsg:4326 (long-lat) and the spherical Mercator projection used by web mapping services. Using epsg codes requires the `rgdal` package to be installed, and `crsLL` is intended as a replacement for `CRS("+init=epsg:4326")` when `rgdal` is not guaranteed to be available.

## Value

Objects of class `CRS`.

## References

[https://en.wikipedia.org/wiki/Web\\_Mercator](https://en.wikipedia.org/wiki/Web_Mercator), <https://spatialreference.org/ref/epsg/4326/>

## See Also

`CRS`, `showEPSG`

## Examples

```
crsMerc
if(require('rgdal', quietly=TRUE))
  CRS("+init=epsg:3857")

crsLL
if(require('rgdal', quietly=TRUE)) {
  CRS("+init=epsg:4326")

  rgdal::showEPSG("+proj=longlat")
  rgdal::showEPSG(as.character(crsLL))
}
```

---

geocode *Georeferencing with Google*

---

## Description

Uses the `dismo` package to geocode with Google

## Usage

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

## Arguments

<code>x</code>	Vector of character strings to search for
<code>extent</code>	Currently unused. an Extent object, or any object from which an Extent can be obtained.
<code>lang</code>	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 retrieved 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(mapmisc::geocode('Ulan batar'), silent=TRUE)  
data('worldMap', package='mapmisc')  
  
if(  
  !all(class(cities) == 'try-error') & requireNamespace("rgdal", quietly = TRUE)  
) {  
  citiesT = spTransform(cities, crs(worldMap))  
  mapmisc::map.new(citiesT, buffer=5000*1000)  
  plot(worldMap, add=TRUE)  
  points(citiesT, col='red')  
  text(citiesT, labels=citiesT$name, col='red',pos=4)  
}
```

---

GNcities	<i>Retrieve city names and locations</i>
----------	--

---

### Description

This function uses the geonames package to provide city names and locations from [www.geonames.org](http://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 <code>codeextend</code>
...	Various search arguments
crs	projection for the output

### 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),xmn=8,xmx=18,ymn=0,ymx=10,
  crs=mapmisc::crsLL)
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)
```

---

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 CRS object, proj4 string, or an object from which a projection can be extracted with <code>proj4string(crs)</code>
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

```

**Author(s)**

Patrick Brown

**See Also**

[gridlines](#), [llgrid](#)

**Examples**

```

## Not run:
Npoints = 20
myPoints = SpatialPointsDataFrame(
  cbind(
    runif(Npoints, -15000000, 15000000),
    runif(Npoints, -8000000, 8000000)),
  data=data.frame(y1=c(NA, rnorm(Npoints-1)),
    y2=c(sample(0:5, Npoints-1,replace=TRUE), NA)),
  proj4string=moll(c(-100,0))
)

plot(myPoints)
gridlinesWrap(myPoints, lty=3, col='red')

## End(Not run)

```

---

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 <a href="#">legend</a> function.
breaks	Optional list with elements col and legend, such as the output from <a href="#">colourScale</a>
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 <a href="#">legend</a>
bg	background colour see <a href="#">legend</a>
cex	see <a href="#">legend</a>
pt.cex	see <a href="#">legend</a>
text.col	see <a href="#">legend</a>
title	see <a href="#">legend</a>
inset	see <a href="#">legend</a>
title.col	see <a href="#">legend</a>
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 <a href="#">legend</a>
...	Additional arguments passed to <a href="#">legend</a> .

### Details

A legend for 'z-axis' colour scales.

### Value

Result of call to [legend](#)

**See Also**[colourScale](#)


---

legendTable	<i>Table for colour scales</i>
-------------	--------------------------------

---

**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 <a href="#">colourScale</a>
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

<code>x</code>	A spatial object from which an extent can be extracted.
<code>legendRight</code>	Leave room to the right for the legend produced by plotting a Raster object
<code>buffer</code>	passed to <a href="#">extend</a> to increase the plotting area
<code>mar</code>	see <a href="#">par</a>
<code>...</code>	Additional arguments passed to <code>plot</code>

## Details

`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=mapmisc::crsLL)

map.new(myPoints, legendRight=TRUE, mar=c(3,3,0,0), buffer=0.2)
points(myPoints)
legendBreaks('right', list(breaks=1:3, col=1:2))
```

---

 modis

*MODIS tiles and projection*


---

## Description

Raster containing MODIS tile ID's

## Usage

```
getModisTiles(x, tiles = mapmisc::modisRaster)
crsModis
modisRaster
degreeRaster
```

## Arguments

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

## Details

getModisTiles returns a matrix with modis tiles.

modisRaster shows horizontal and vertical tile names for downloading data from MODIS at, for example, <http://e4ftl01.cr.usgs.gov/MOTA>.

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

## References

[https://nsidc.org/data/docs/daac/mod10\\_modis\\_snow/landgrid.html](https://nsidc.org/data/docs/daac/mod10_modis_snow/landgrid.html), <https://spatialreference.org/ref/sr-org/modis-sinusoidal/>

## Examples

```
crsModis

myPointLL = SpatialPoints(cbind(c(5:6),10:11), proj4string = crsLL)

if(requireNamespace("rgdal", quietly=TRUE)) {
  getModisTiles(myPointLL)

  getModisTiles(myPointLL, mapmisc::degreeRaster)
}

modisUrl = 'ftp://ladsweb.nascom.nasa.gov/allData/5/MCD12Q1/2002/001/'

## Not run:
paste(modisUrl,
```

```
grep(
  paste(getModisTiles(myPointLL)[,'tile'], collapse='|'),
  unlist(strsplit(RCurl::getURL(
    modisUrl,ftp.use.epsv=TRUE,
    dirlistonly = TRUE), '\n')),
  value=TRUE), sep='')

## End(Not run)
```

---

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 `nldCities` is a `SpatialPoints-DataFrame` 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

[meuse](#), [getData](#), [openmap](#)

## Examples

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

# proj4string(meuse) <- CRS("+init=epsg:28992")
proj4string(meuse) <- CRS(
  paste("+proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763888888889",
```

```

      "+k=0.9999079 +x_0=155000 +y_0=463000 +ellps=bessel +units=m",
      "+no_defs +datum=WGS84")
    )

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)
if(requireNamespace('rgdal', quietly=TRUE)) 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, mapmisc::crsLL)
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))
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=~ /research/diseasemapping/pkg/mapmisc/data/netherlands.RData",
# compress="xz")

## End(Not run)

```



**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)
ocea(x, angle, flip=FALSE)
moll(x=0, angle=NULL, flip=FALSE)
```

**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
flip	post-projection flipping of coordinates
preserve	A SpatialPoints object, the resulting projection is scaled so as to preserve the distances between these points as best as possible.

**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/wiki/Space-oblique\\_Mercator\\_projection/](https://en.wikipedia.org/wiki/Space-oblique_Mercator_projection/)

**See Also**

[CRS-class,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=5))
  )
  projection(nldRot)

  map.new(nldRot)
  text(nldRot, labels=nldRot$name)
  scaleBar(nldRot, 'topright')
  insetMap(nldRot, 'bottomright', map=world)
}
## Not run:

nldOceaCrs = ocea(nldCities)
nldOcea = spTransform(nldCities, nldOceaCrs)
map.new(nldOcea, buffer=2000)
text(nldOcea, labels=nldOcea$name)
scaleBar(nldOcea, 'topright')

map.new(nldCities)
plot(nldTiles, add=TRUE)
text(nldCities, labels=nldCities$name)

tilesRot = openmap(nldRot)
map.new(nldRot)
plot(tilesRot, add=TRUE)
text(nldRot, labels=nldRot$name)

tilesUtm = openmap(nldUtm)
map.new(nldUtm)
plot(tilesUtm, add=TRUE)
text(nldUtm, labels=nldUtm$name)

## End(Not run)

```

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=raster::crs(x),
  buffer=0, fact=1,
  verbose=getOption('mapmiscVerbose'),
  cachePath=getOption('mapmiscCachePath')
)

osmTiles(name, xyz, suffix)

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

**Arguments**

x	An <b>extent</b> 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	Source of map tiles, see <a href="http://diseasemapping.r-forge.r-project.org/mapLayers.html">http://diseasemapping.r-forge.r-project.org/mapLayers.html</a> .
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 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 <b>disaggregate</b> before reprojecting if fact>1, helps 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 <http://diseasemapping.r-forge.r-project.org/mapLayers.html> for a full set of map tiles.

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 Raster with indexed colours, or a RasterBrick `brick`, with 'red', 'green' and 'blue' layers.

`openmapAttribution` returns a character string.

### Examples

```
data("netherlands")

plot(nldTiles)

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

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

myraster = raster(matrix(0,10,10),xmn=8,xmx=18,ymn=0,ymx=10,
  crs=mapmisc::crsLL)

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, zoom=6)
```

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

mytiles = openmap(myPoints,
path='waze', verbose=TRUE)
map.new(myPoints)
plotRGB(mytiles, add=TRUE)
points(myPoints, col='red')
openmapAttribution(mytiles)

## End(Not run)
```

---

persistentCache	<i>Set a persistent cache</i>
-----------------	-------------------------------

---

## 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.

## 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
## Not run:
persistentCache(verbose=TRUE)
```

```
## End(Not run)
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 = 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 <code>proj4string(crs)</code>
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 <code>openmap</code> 's path argument
zoom	Zoom level if retrieving inset map from <code>openmap</code>
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 <code>polygon</code> )
cropInset	Crop the insert map to this extent
...	Additional arguments passed to <code>legend</code> for <code>scaleBar</code> or <code>polygon</code> (for <code>insetMap</code> ).

## 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.

## 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=mapmisc::crsLL)

## 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, seg.len=2)
temp=legendBreaks("topleft", legend=breaks, col=thecol)

## Not run:
thedot = insetMap(myPoints, "bottomleft",col='#00000000', lty=0)
points(thedot)

## End(Not run)
```

---

tonerToTrans	<i>Convert RGB maps to semi-transparent</i>
--------------	---

---

### 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)$", power = 0.5, col='black', threshold=Inf)
rgbtToIndex(x, pattern="(red|green|blue|trans)$")
```

### Arguments

x	A RasterStack with RGB colours, such as from <a href="#">openmap</a> with path='stamen-toner'
pattern	string passed to <a href="#">grep</a> 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

### Details

The difference between these functions is that tonerToTrans converts white to transparent, whereas rgbtToIndex uses the transparency layer. The former is intended for 'stamen-toner' maps.

### Value

A RasterLayer with indexed colours

### Author(s)

Patrick Brown

### See Also

[openmap](#)

### Examples

```
## Not run:

rgbMap = openmap(c(0,10), zoom=3, path='stamen-toner')
names(rgbMap)
plotRGB(rgbMap)
```



```

transMap = tonerToTrans(rgbMap, col='blue')
names(transMap)
par(bg='red')
plot(transMap)

rgbMap[['stamen.tonerTrans']] = 255-rgbMap[['stamen.tonerRed']]
rgbtMap = rgbtToIndex(rgbMap)
plot(rgbtMap)

## End(Not run)

```

---

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, 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
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

An object of class [CRS](#).

## References

[https://en.wikipedia.org/wiki/Two-point\\_equidistant\\_projection/](https://en.wikipedia.org/wiki/Two-point_equidistant_projection/) <https://proj.org/operations/projections/tpers.html>

## 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')
}
```

---

 worldMap

*Country boundaries*


---

**Description**

Country borders from [naturalearthdata.com](http://naturalearthdata.com)

**Usage**

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

**Source**

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

**Examples**

```
# soil data
library("mapmisc")

data("worldMap")
myMap = openmap(worldMap, zoom=1, fact=1.5)
map.new(worldMap)
plot(myMap, add=TRUE)
plot(worldMap, border='red', lwd=3, add=TRUE)
plot(worldMap[worldMap$NAME == 'Brazil',],
      add=TRUE, col='green')
```

---

wrapPoly	<i>Reproject polygons with wrapping</i>
----------	---

---

**Description**

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

**Usage**

```
wrapPoly(x, crs)
```

**Arguments**

x	A Spatial object
crs	An object of class <a href="#">CRS</a> .

**Value**

A reprojected Spatial object.

**See Also**

[spTransform](#).

# Index

- \* **datasets**
  - netherlands, [15](#)
  - worldMap, [26](#)
- breaksForRates (colourScale), [3](#)
- brewer.pal, [3](#)
- brick, [20](#)
- classIntervals, [3, 4](#)
- col2html, [2](#)
- col2rgb, [2](#)
- colorScale (colourScale), [3](#)
- colours, [2](#)
- colourScale, [3, 11, 12](#)
- CRS, [6, 17, 25, 27](#)
- crsLL (crsMerc), [6](#)
- crsMerc, [6](#)
- crsModis (modis), [14](#)
- degreeRaster (modis), [14](#)
- disaggregate, [19](#)
- extend, [8, 13](#)
- extent, [19](#)
- geocode, [7](#)
- getData, [15](#)
- getModisTiles (modis), [14](#)
- GNCities, [8, 8](#)
- GNsearch, [8](#)
- GNsearch (GNCities), [8](#)
- grep, [24](#)
- gridlines, [10](#)
- gridlinesWrap, [9](#)
- hexmode, [2](#)
- insetMap (scaleBar), [22](#)
- legend, [11, 22](#)
- legendBreaks, [4, 10](#)
- legendTable, [12](#)
- llgrid, [10](#)
- map.new, [13](#)
- mapmiscCache (persistentCache), [21](#)
- mapmiscCachePath (persistentCache), [21](#)
- meuse, [15](#)
- modis, [14](#)
- modisRaster (modis), [14](#)
- moll (omerc), [16](#)
- netherlands, [15](#)
- nldCities (netherlands), [15](#)
- nldElev (netherlands), [15](#)
- nldTiles (netherlands), [15](#)
- oceania (omerc), [16](#)
- omerc, [16](#)
- openmap, [15, 19, 22, 24](#)
- openmapAttribution (openmap), [19](#)
- osmTiles (openmap), [19](#)
- par, [13](#)
- persistentCache, [21](#)
- polygon, [22](#)
- rgb, [2](#)
- rgbtToIndex (tonerToTrans), [24](#)
- scaleBar, [4, 22](#)
- scalebar, [13, 23](#)
- showEPSG, [6](#)
- spplot, [13, 23](#)
- spTransform, [17, 25, 27](#)
- tonerToTrans, [24](#)
- tpeqd, [25](#)
- tpers (tpeqd), [25](#)
- world (netherlands), [15](#)
- worldMap, [26](#)
- worldmap (worldMap), [26](#)
- wrapPoly, [27](#)