The mapmisc package

Patrick Brown

February 26, 2014

1 Introduction

This package provides a few utilities for making nice maps in R, with an emphasis on enabling maps in short tidy code chunks which are suitable for Sweave and knitr documents. The package duplicates the capabilities of packages such as classInt, geonames, and OpenStreetMap, and the price of having tidier code is much of the flexibility from these other packages has been lost here.

The meuse data from the sp package

```
library("sp")
library("rgdal")
data("meuse")
coordinates(meuse) <- ~x + y
proj4string(meuse) <- CRS("+init=epsg:28992")
class(meuse)

## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

Get elevation data with

```
meuseLL = spTransform(meuse, CRS("+init=epsg:4326"))
library("raster")
getData("SRTM", lon = xmin(extent(meuseLL)), lat = ymin(extent(meuseLL)), path = tempdir
nldElev = raster(paste(tempdir(), "/", "srtm_38_02.tif", sep = ""))
nldElev = crop(nldElev, extent(meuseLL))
nldElev = projectRaster(nldElev, crs = proj4string(meuse))
```

Or simply do

```
library("mapmisc")
data("netherlands")
```

The elevation data is a Raster.

```
class(nldElev)

## [1] "RasterLayer"
## attr(,"package")
## [1] "raster"

nldElev = crop(nldElev, extend(extent(meuse), 1000))
```

2 Downloading background maps and city locations

Get a background map covering the extent of the meuse data

```
library("mapmisc")
nldTiles = openmap(meuse)
```

nldTiles is a Raster with the same projection as meuse

```
class(nldTiles)

## [1] "RasterLayer"
## attr(,"package")
## [1] "raster"

proj4string(nldTiles)

## [1] "+init=epsg:28992 +proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763888888889 +

proj4string(meuse)

## [1] "+init=epsg:28992 +proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763888888889 +
```

Maps which can be downloaded are shown at `http://diseasemapping.r-forge.r-project.org/openmap/`

GNcities is a wrapper for the function of the same name in the geonames package.

```
nldCities = GNcities(meuse, maxRows = 6)
```

A SpatialPointsDataFrame, with same map projection.

```
class(nldCities)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"

names(nldCities)

##  [1] "fcodeName"   "toponymName" "countrycode" "fcl"         "fclName"
##  [6] "name"        "wikipedia"   "lng"         "fcode"       "geonameId"
## [11] "lat"         "population"

proj4string(nldCities)

## [1] "+init=epsg:28992 +proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763888888889 +
```

3   Making maps

The map.new function sets up a map in the current plot window with the correct limits and aspect ratio for the object supplied, and without margins or white space. scaleBar adds a scale and north arrow. It uses the map projection of the argument supplied to calculate distances and find north.

```r
# plot the data locations
map.new(meuse)
plot(nldTiles, add = TRUE)
points(meuse, col = "red", cex = 0.3)
scaleBar(proj4string(meuse), pos = "topleft", bg = "white")

# plot city names
map.new(meuse)
plot(nldTiles, add = TRUE)
points(nldCities)
text(nldCities, labels = nldCities$name, pos = 3)
scaleBar(proj4string(meuse), pos = "topleft", bg = "white")

# plot elevation
map.new(meuse, legendRight = TRUE)
plot(nldTiles, add = TRUE)
plot(nldElev, add = TRUE, col = terrain.colors(8), alpha = 0.6, legend.mar = 2,
    legend.line = 0, )
scaleBar(meuse, pos = "topleft", bg = "white")
```

(a) data locations   (b) cities   (c) elevation

Figure 1: simple map

Reproject the data so it's roughly aligned with the x-axis.

```
rotatedCRS = CRS("+proj=omerc +lat_0=51 +lonc=5.75 +alpha=-50.0000000 +gamma=0.0 +k=1.00
meuseRot = spTransform(meuse, rotatedCRS)
tilesRot = projectRaster(nldTiles, crs = rotatedCRS, method = "ngb")
tilesRot@legend@colortable = nldTiles@legend@colortable
elevRot = projectRaster(nldElev, crs = rotatedCRS)
```

And create new plots

```
# first elevation
map.new(meuseRot)
plot(tilesRot, add = TRUE)
plot(elevRot, add = TRUE, alpha = 0.5, col = terrain.colors(8), legend = FALSE)
points(nldCities)
text(nldCities, labels = nldCities$name, pos = 3)
scaleBar(meuseRot, pos = "topleft", bg = "white")

# then data locations
map.new(meuseRot)
plot(tilesRot, add = TRUE)
points(meuseRot, col = "red", cex = 0.3)
scaleBar(proj4string(meuseRot), bg = "white")
```

4
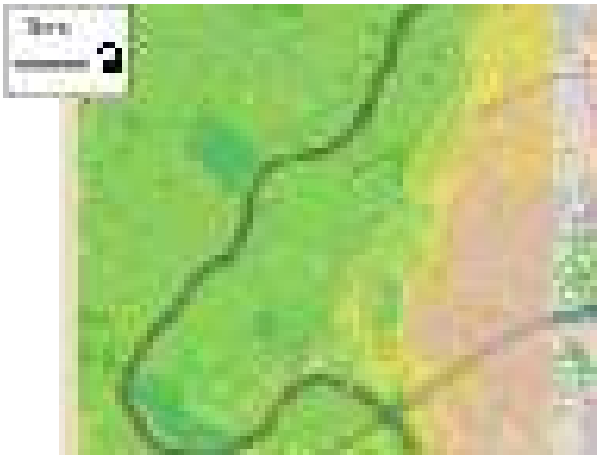
(a) elevation

(b) data

Figure 2: Rotated map

4 Inset maps

```r
world = openmap(extent(-10, 30, 40, 60), path = "osm-no-labels")
```

```r
# not rotated
map.new(meuse, legendRight = TRUE)
plot(nldTiles, add = TRUE)
points(meuse)
scaleBar(proj4string(meuse), pos = "bottomright", bg = "white")
insetMap(meuse, "topright", map = world)


# rotated
map.new(meuseRot)
plot(tilesRot, add = TRUE)
points(meuseRot, col = "red", cex = 0.3)
scaleBar(proj4string(meuseRot), bg = "white")
insetMap(meuseRot, "bottomleft", map = world)
```

<div style="text-align:center">

(a) elevation   (b) rotated

Figure 3: Inset map

</div>

5   Legends

Create a colour scale for plotting copper concentrations

```
cuScale = colourScale(meuse$copper, breaks = 5, style = "equal", opacity = 0.8,
    dec = -1, firstBreak = 0)
```

and elevation, with transparency decreasing as elevation increases.

```
elevScale = colourScale(nldElev, style = "equal", breaks = 6, col = terrain.colors,
    firstBreak = 0, dec = -1, opacity = c(0.2, 0.9))
```

Soil type is a categorical variable, create a factor and create a colour scale of unique values

```
meuse$soilFac = factor(meuse$soil, levels = c(1, 2, 3), labels = c("Calcareous",
    "Non-Calc's", "Red Brick"))
soilScale = colourScale(meuse$soilFac, col = "Set2")
```

```
map.new(meuse)
plot(nldTiles, add = TRUE)
plot(meuse, col = cuScale$plot, add = TRUE, pch = 16)
scaleBar(meuse, pos = "topleft", bg = "white")
legendBreaks("bottomright", breaks = cuScale, title = "gals/firkin")
```

<div style="text-align:center">

6

</div>

```
map.new(meuse)
plot(nldTiles, add = TRUE)
plot(meuse, col = soilScale$plot, add = TRUE, pch = 16)
scaleBar(meuse, pos = "topleft", bg = "white")
legendBreaks("bottomright", breaks = soilScale, title = "soil type", cex = 0.7,
    bg = "white")

map.new(meuse)
plot(nldTiles, add = TRUE)
image(nldElev, breaks = elevScale$breaks, col = elevScale$colOpacity, legend = FALSE,
    add = TRUE)
scaleBar(meuse, pos = "topleft", bg = "white")
legendBreaks("left", breaks = elevScale, title = "Metres", bg = "white")
```


(a) Copper   (b) soil   (c) elevation

Figure 4: Meuse data again