Creating NUTS-2 Choropletes from Eurostat Data

Albrecht Gradmann

March 5th, 2015

Introduction

This document shows how to create a choroplete map from Eurostat Data using R. For this example I aimed to automate the entire process from downloading the data to the creation of the figure. The idea is to present an entirely reproducible process. The text shows the overall process from a birds-eye perspective. In particular it is not intended to serve as a general introdution to R[1] or a discussion of the Pros and cons of using choropletes to display data. In addition I plan to create other handouts in the future. Please let me know if you would like to request a specific topic.

[1] One of many good introductory sites for R. `http://www.statmethods.net/`

Setting the scene

The example relies on a series of libraries which have to be installed on your system. Package installation in R is a simple and standard task and it can be done in several ways. For more information, you can either type `help("install.packages")` in the R Console or look for one of the numerous tutorials on the internet[2]. Below is the series of commands you need to run first to attach the packages to your R session.

[2] `http://www.dummies.com/how-to/content/how-to-install-load-and-unload-packages-in-r.html`

```
library(maptools) # Dealing with spatial data
library(ggplot2)  # Hadley Wickham plotting package
library(ggmap)    # Mapping with ggplot
library(RJSDMX)   # Query Eurostat REST-interface
library(grid)     # Needed for unit() function
```

Getting the unemployment data

Use the `getTimeSeries()` function to download the dataset from Eurostat. R provides some facilities to browse the Eurostat database. The details of these facilities and an explanation how to define the arguments will be topic of another handout. The code below basically tells R to got to the Eurostat website, look for a dataset called `tgs00010` and from this to select annual data (that's the `A`), measured as percentage (the `PC` part) for both sexes (`T` stands for total) and for all available geographic entities (`*` means 'take all'). The `start` and `stop` arguments define the timespan for which we want to get the data.

```
# Get the data formatted as list from Eurostat
tsList = getTimeSeries('EUROSTAT',
                       'tgs00010.A.PC.T.*',
                       start = "2013",
                       end="2013")
```

The next step is to convert the downloaded data into a format suitable for further processing. The format is basically a table of different variables called a `data-frame` in R. The command `head` allows you to inspect the first few lines of the downloaded data. If you have ever worked with Eurostat data you will probably recognize the

```
# Convert the list into a dataframe
tsDf <- sdmxdf(tsList,meta = T)
```

	TIME	OBS	FREQ	UNIT	SEX	GEO
1	2013	4.00	A	PC	T	AT11
2	2013	4.50	A	PC	T	AT12
3	2013	8.40	A	PC	T	AT13
4	2013	5.30	A	PC	T	AT21
5	2013	4.00	A	PC	T	AT22
6	2013	4.00	A	PC	T	AT31

Table 1: First rows of tsDf

```
# Subset to exclude Turkey
tsDfsub <-  subset(tsDf,
                   !grepl(c("TR"),GEO))
```

Getting the empty maps

The next step will be to read the data of administrative boundaries into R. For this Eurostat provides ESRI-shapefiles [3].

[3] `http://ec.europa.eu/eurostat/web/gisco/geodata/reference-data/administrative-units-statistical-units`

```
# Get the shapefile from Eurostat
download.file("http://ec.europa.eu/eurostat/cache/GISCO/geodatafiles/NUTS_2010_60M_SH.zip",
              paste0(tempdir(),
                     "/NUTS_2010_60M_SH.zip"))

# Unzip the data in a temporary location
unzip(paste0(tempdir(),
             "/NUTS_2010_60M_SH.zip"),
      exdir=tempdir())

# Read administrative boundaries
```

```
eurMap <- readShapePoly(fn=paste0(tempdir(),
      "/NUTS_2010_60M_SH/NUTS_2010_60M_SH/data/NUTS_RG_60M_2010"))

# And convert it into a format suitable for plotting
# which is again a dataframe
eurMapDf <- fortify(eurMap, region="NUTS_ID")
```

Merge Unemployment Data with Map Data

```
# merge map and data
tsMapDf <- merge(eurMapDf, tsDfsub, by.x="id", by.y="GEO")
tsMapDf <- tsMapDf[order(tsMapDf$order),] ##crucial step!
```

Explain that the polygons are printed in the order given in the table, and that it is crucial to put them in correct order first.

Plotting the data

Now we are ready to plot the data.

```
# inverse order (to have visible borders)
map <- ggplot(data = tsMapDf, aes(x = long, y = lat,
    group = group))
map <- map + geom_polygon(aes(fill = OBS))  #+  coord_equal()
map
```

Obviously there are several problems with this display, and it is certainly not useful for publishing. However, at this stage we may want to check if the results are generally sensible. We will

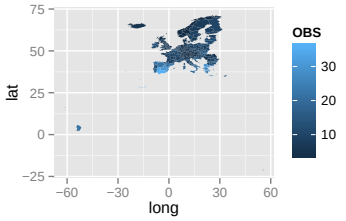


Figure 1: An initial version of our map

Tuning the plot

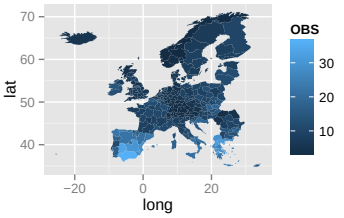
Putting Europe into focus

```
#limit data to main Europe
europe.limits <- geocode(c("Cape Fligely, Rudolf Island,
                           Franz Josef Land, Russia",
                           "Gavdos, Greece", "Faja Grande,Azores",
                           "Severny Island, Novaya Zemlya, Russia"))

# apply the limits to our dataset
tsMapDf <- subset(tsMapDf,
                        long > min(europe.limits$lon) &
                          long < max(europe.limits$lon) &
                          lat > min(europe.limits$lat) &
```

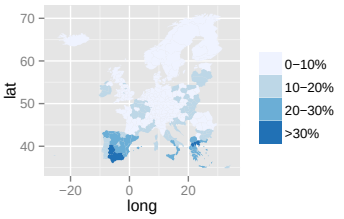
```
                        lat < max(europe.limits$lat))

# and re-read the plot with the new data
map <- map %+% tsMapDf
map
```



Bin data into classes

```
map <- ggplot(data=tsMapDf,aes(x=long, y=lat, group=group))
map <- map + geom_polygon(aes(fill=cut(OBS, breaks=c(0,10,20,30,100))))
map <- map + scale_fill_brewer(name="",labels=c("0-10%","10-20%","
                               guide="legend")
map
```



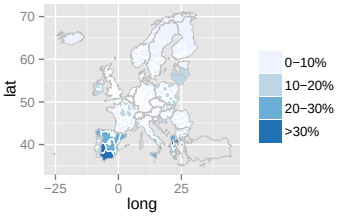
Add country borders

```
# Borders on NUTS2 level in white
map <- map + geom_path(color="white",size=0)

# Borders on NUTS1 level
rindex <- grep(eurMapDf$id,pattern=c("^[[:alpha:]]{2}$"))
eurMapDf_NUTS1 <- eurMapDf[rindex,]

eurMapDf_NUTS1 <- subset(eurMapDf_NUTS1,
                         long > min(europe.limits$lon) &
                           long < max(europe.limits$lon) &
                           lat > min(europe.limits$lat) &
                           lat < max(europe.limits$lat))

map + geom_path(data=eurMapDf_NUTS1, color='grey', size=0.1)
```



Some window dressing

```
map <- map + theme_bw()
map <- map + theme(
    plot.background = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.border = element_blank(),
    axis.ticks = element_blank(),
    axis.text = element_blank(),
    axis.title = element_blank(),
    legend.key.size = unit(10,"pt"),
```

```
    text=element_text(size=8),
    legend.position=c(0.1, 0.2))
```

The Result

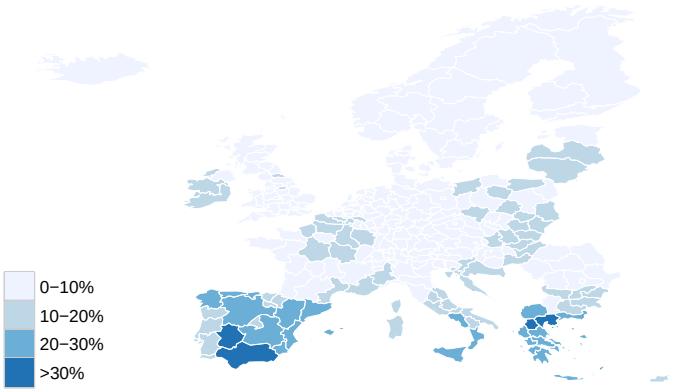


Figure 2: EU 2013 Unemployment rates by NUTS2 regions, percentage of total population, both sexes Source: Eurostat (tgs00010)

Concluding remarks and Credits

Tufte[4] For this [5]

Stackoverflow[6] Max Marchis Blog[7] Announcement or eurostat package[8] ggplot2 documentation[9]

Sidenotes

One of the most prominent and distinctive features of this style is the extensive use of sidenotes. There is a wide margin to provide ample room for sidenotes and small figures. Any use of a footnote will automatically be converted to a sidenote. [10]

If you'd like to place ancillary information in the margin without the sidenote mark (the superscript number), you can use the `\marginnote` command.

[4] `http://www.edwardtufte.com/tufte/books_be`

[5] `https://code.google.com/p/tufte-latex/`

[6] `http://stackoverflow.com/`

[7] `http://www.milanor.net/blog/?p=594`

[8] `http://rstudio-pubs-static.s3.amazonaws.com/27120_4dea44a84c9247c797289e145c17b38d.html`

[9] Online Documentation of Hadley Wickhams `ggplot2` package. `http://docs.ggplot2.org/current/`

[10] This is a sidenote that was entered using a footnote.

This is a margin note. Notice that there isn't a number preceding the note.