SUPPLEMENT 1: DRAWING MAPS

7316 - INTRODUCTION TO DATA ANALYSIS WITH R

Mickaël Buffart (mickael.buffart@hhs.se)

This short document is a supplement to the module on graphs with `ggplot`, to help you draw static maps with R. Going through this material is optional, depending on your own interest. R also allows to create interactive maps. We will cover interactive graphs, including maps, in the final module.

To go further than this short document, the book in the course litterature on *Elegant Graphics for Data Analysis*, from Hadley Wickam, has a section on maps.

1. Maps are graphs

Static maps are a form of graphs, with the following properties:

- The x and y axis are often hidden. In ggplot2, you can use `theme_void()` to hide the axes.
- The x and y axis are used as projection of geographical coordinates: latitude and longitude. Because the earth is a sphere, the choice of projection you do will change the appearance of your map.
- city, region, or country border are usually depicted with polygons. Internet is full of geographical polygon databases that you can use as a background layer of your maps. One nice source of polygon data is *Natural Earth*, integrated in R with the package `rnaturalearth`.

Once you chose your polygon database for the part of the globe you wish to project, you may add other layers to your graph as you usually do with ggplot, using `geom_point()` for dots at specific latitudes and longitudes, `geom_sf()` to fill surfaces (*polygons*) with colors, or `geom_sf_label()` and `geom_sf_text()` to add labels and texts to your map.

2. Example: plotting Europe

1. First, we want to load and plot the polygons of Europe as a background layer.

```
# Get country polygon from the rnaturalearth package
world <- rnaturalearth::ne_countries(scale = "medium",
                                     returnclass = "sf")

# we want to draw a map of Europe. Let's subset this part of the world
Europe <- world[world$continent == "Europe",]
```

- In the chunk of code above, `rnaturalearth::ne_countries()` create a spatial polygons dataframe, containing the country polygons of the world.
  - The `scale` indicate the zoom level of the map.
  - the `returnclass` provide the spatial representation you would like to create. It provides two options: `sp`, or `sf`. I suggest you always use `sf`, as `sp` will be deprecated in future versions.

2. Now, you have enough to draw your map with ggplot.

```
# Load ggplot2
library(ggplot2)
```

```
Warning: package 'ggplot2' was built under R version 4.3.1
```

```
# We use the Europe dataset that we loaded
map_1 <- ggplot(data = Europe) +
  # This is a spatial "sf" representation
  geom_sf() +
  # We want to zoom the map on Europe latitude and longitude
  coord_sf(xlim = c(-20, 40),
           ylim = c(35, 70))

# We plot the map
map_1
```

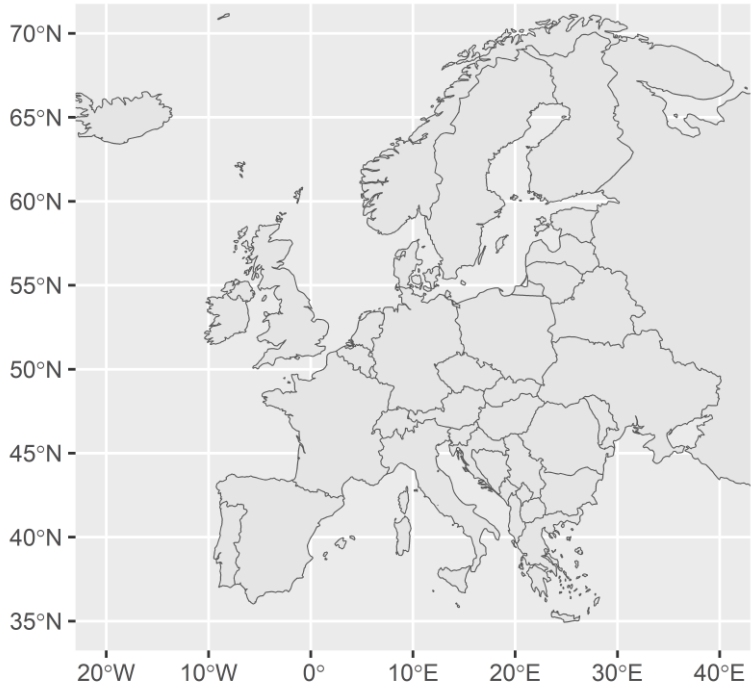

Figure 1: A simple map with polygons

- In the example above, we simply plot the country polygons that we loaded above. We can remove the coordinates background using `theme_void()`.

```
# We remove the axes from the map
map_1 <- map_1 + theme_void()
```

```
# Display the map
map_1
```

Figure 2: The map has no axes

Now, we have a template of Europe with gray colored countries. We may want to add colors and dots.

3. Let's add dots to represent the cities of Stockholm and Göteborg.

```
# Let's create a dataframe containing the coordinates of Stockholm and Göteborg
cities <- data.frame(city = c("Stockholm", "Göteborg"),
                     latitude  = c(59.334591, 57.708870),
                     longitude  = c(18.063240, 11.974560),
                     stringsAsFactors = FALSE)

# Let's add the points to the map
map_1 <- map_1 +
  geom_point(data = cities,
             mapping = aes(x = longitude, y = latitude),
             colour = "#EC3D2F",
             size = 3)

# Display the map
map_1
```

Figure 3: We add points to the map

- In the code above, we add `geom_point()` to the `map_1` graph. The data for the position of the points are in the dataframe cities. `x` corresponds to the `longitude` variable, and `y` to the `latitude`.
- We also choose a color and a size for the points. This is optional.

4. Now, let's imagine that you would like to color the country based on some country characteristics: level of hapiness, GDP, etc. you may simply add a column in the polygon dataset containing the value of the variable you want to depict.

WARNING: make sure your country (or city, or region) identifiers are spellt exactly the same as in the polygon dataset. Otherwise the `merge()` will create missing values.

```
# We have a country dataset measure the level of levelness
mydf <- data.frame(country   = c("Albania", "Austria", "Belgium", "Estonia",
                                 "Finland", "France", "Faeroe Is.",
                                 "United Kingdom", "Greece", "Sweden"),
                   levelness = c(1, 3, 4, 2, 6, 1, 2, 1, 3, 7))

# We merge mydf with the Europe polygon dataset
Europe <- merge(x = Europe, y = mydf,
                by.x = "name", by.y = "country",
                all.x = TRUE, all.y = FALSE)

# Warning: if a country is missing in our levelness dataset,
#  we still want to keep the polygon (hence, all.x = TRUE, all.y = FALSE)
```

5. Now, we can color the polygon with gradient

```
# We updated the Europe dataset. Let's reload the data in the graph
map_2 <- ggplot(data = Europe) +
  geom_sf() +
  coord_sf(xlim = c(-20, 40),
           ylim = c(35, 70)) +
  theme_void()

# Adding aestetics with filling on the map_2
map_2 <- map_2 + aes(fill = levelness)

# Display the map
map_2
```

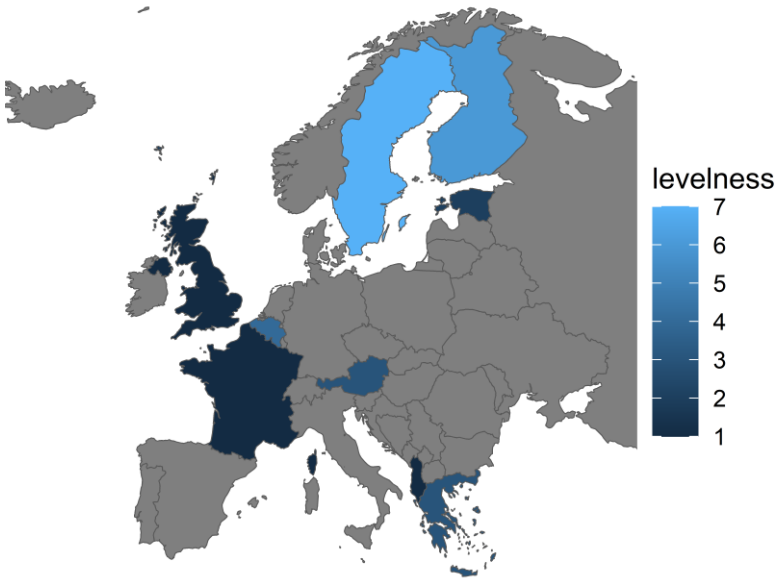

Figure 4: We add colors to the map

Remember that the variables defining the colors of the polygons need to be added to the polygon dataframe before the map is created. Otherwise, R will tell you that the variable does not exist.

You can then adjust the colors and choice of variable as you like, following what we have seen for ggplot2 customization during the module.