campR – visualization of geolocations

Marcus Vollmer

2020-05-20

Contents

This is a campR notebook introducing the visualization of geolocations on an open source map.

Libraries and data

The following commands imports neccessary libraries.

```
library(RCurl)

library(ggplot2)
library(ggmap)
library(ggrepel)
library(mapproj)

library(rworldmap)
```

We are going to use the locations of world-wide earthquakes that can be imported from an in-build data set. Geographical coordinates are usually stored as a pair of latitudinal (**lat**) and longitudinal (**long**) values.

```
data(quakes)
head(quakes)
```

```
##      lat   long depth mag stations
## 1 -20.42 181.62   562 4.8       41
## 2 -20.62 181.03   650 4.2       15
## 3 -26.00 184.10    42 5.4       43
## 4 -17.97 181.66   626 4.1       19
## 5 -20.42 181.96   649 4.0       11
## 6 -19.68 184.31   195 4.0       12
```

Furthermore, you might have named locations of the study centre or patient residencies stored in a column of your dataset. Lets have look at this dummy data set with 20 patients included from a multicentre study:

```
 D <- data.frame(ID=1:20, Centre=c(rep("Greifswald",1,10), rep("Stralsund",1,3), rep("Berlin",1,5), rep
summary(D)
```

```
##        ID               Centre
##  Min.   : 1.00   Berlin    : 5
```

```
##  1st Qu.: 5.75   Greifswald:10
##  Median :10.50   Hamburg   : 2
##  Mean   :10.50   Stralsund : 3
##  3rd Qu.:15.25
##  Max.   :20.00
```

How to get geographical locations

For these named locations you need to search for **lat** and **long** coordinates to project some information on a geographical map. There are free but limited services available, e.g., MapQuest. If you have a registered account you can use the API to automatically extract geolocations. The **RCurl** package is needed to perform HTTP requests.

```
mq_location <- function(location, KEY) {
  url <- paste0('http://www.mapquestapi.com/geocoding/v1/address?key=', KEY, '&outFormat=csv', '&loca
  data <- read.csv(textConnection(getURL(url), encoding="UTF-8"), header=TRUE)
  return(data)
}
```

To extract the coordinates of Greifswald (Germany) use your own user **key** from MapQuest.

```
KEY <- 'ENTERYOURKEYHERE'
```

```
location = "Greifswald, Germany"
myloc = mq_location(location, KEY)
myloc
```

```
##   Country                   State County       City PostalCode Street      Lat
## 1      DE MECKLENBURG-VORPOMMERN     NA Greifswald         NA     NA 54.09579
##        Lng DragPoint    LinkId Type GeocodeQualityCode GeocodeQuality
## 1 13.38152     false 282245742    s               A5XAX           CITY
##   SideOfStreet DisplayLat DisplayLng
## 1            N   54.09579   13.38152
```

The function may return multiple findings which you should check carefully. Or you could use the mean coordinate for quick assignment of the coordinate.

```
c(mean(myloc$Lat), mean(myloc$Lng))
```

```
## [1] 54.09579 13.38152
```

Lets find the coordinates of our dataset **D** by looping the different places. To avoid false locations you can add the state while searching. Here we save the results in a separate data frame.

```
L <- data.frame(place=levels(D$Centre), lat=0, lng=0, size=0)
for (i in 1:NROW(L)) {
  myloc <- mq_location(paste0(L$place[i],', Germany'), KEY)
  L$lat[i]  <- median(myloc$Lat)
  L$lng[i]  <- median(myloc$Lng)
  L$size[i] <- sum(D$Centre==L$place[i])
}
```

Alternatively, you may also use the Google API which also requires a user key.

```
library(ggmap)
register_google(key="[your key]")
mylocs <- geocode(paste0(locations, ", Germany"), source="google")
mylocs
```

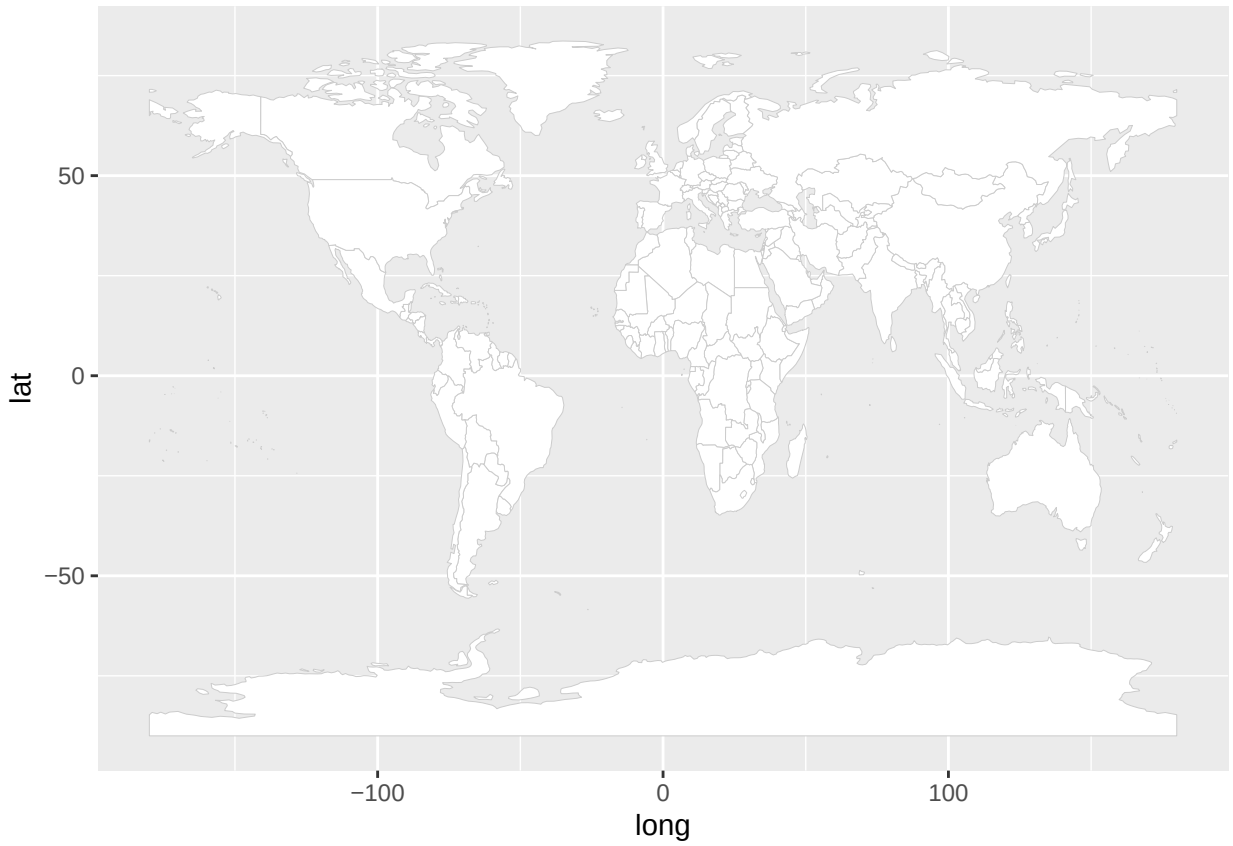
Get a free map as an underlying layer for ggplot

Maps usally store borders of countries, states, cities, rivers, lakes, etc. as polygones and are available from at different resolution. E.g., the R package **rworldmap** contains the borders of all states. You can use the plot or ggplot command to draw the polygons.

```
# Get the world map
  worldMap <- getMap()
  plot(worldMap, col='lightgrey', border='darkgrey')
```

```
  ggplot() +
    geom_polygon(data=worldMap, aes(long, lat, group=group), colour='grey80', size=0.1, fill="white")
```

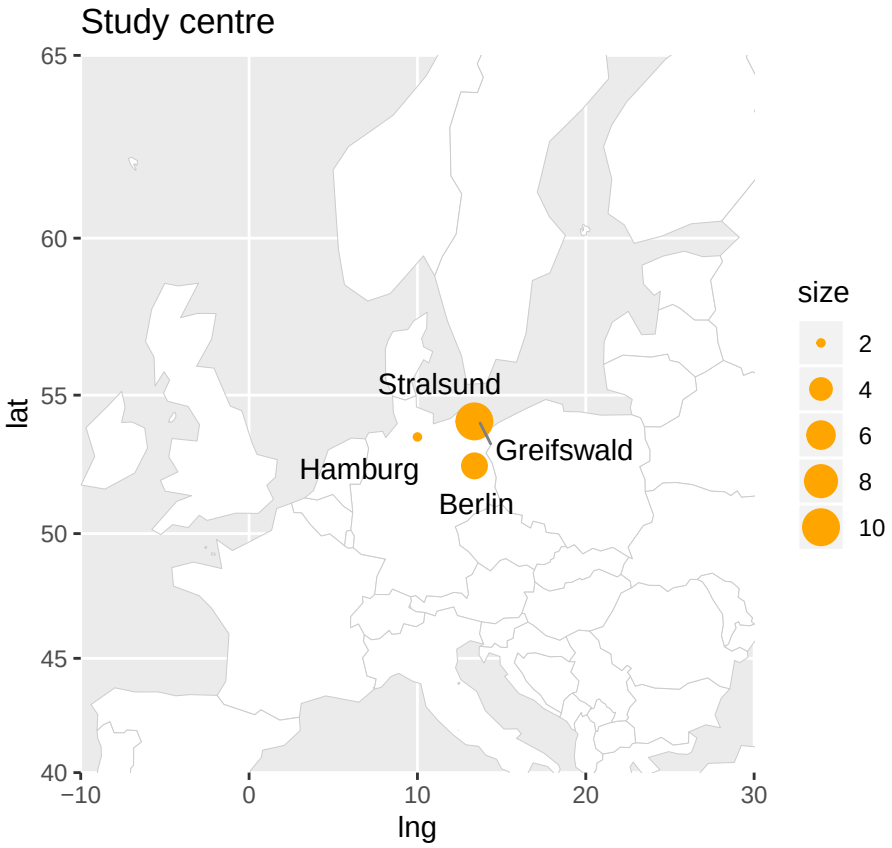
```
## Regions defined for each Polygons
```



You have also the opportunity to extract longitude and latitude borders coordinates of all states for editing purposes. Here we plot the study center of our dummy data frame and set the limits of the axis by **coord_map**.

```
  All = 1:length(worldMap$NAME)
  Coords = lapply(All, function(i){
    df <- data.frame(worldMap@polygons[[i]]@Polygons[[1]]@coords)
    df$region = as.character(worldMap$NAME[i])
    colnames(df) = list("lng", "lat", "region")
    return(df)
  })
  Coords <- do.call("rbind", Coords)

ggplot() +
  geom_polygon(data=Coords, aes(x=lng, y=lat, group=region), colour='grey80', size=0.1, fill="white") +
  coord_map(xlim=c(-10, 30), ylim=c(40, 65)) +
  geom_point(data=L, aes(x=lng, y=lat, size=size), color="orange") +
  geom_text_repel(data=L, aes(x=lng, y=lat, label=place), color='black', box.padding=unit(0.25, "lines"
  ggtitle("Study centre")
```

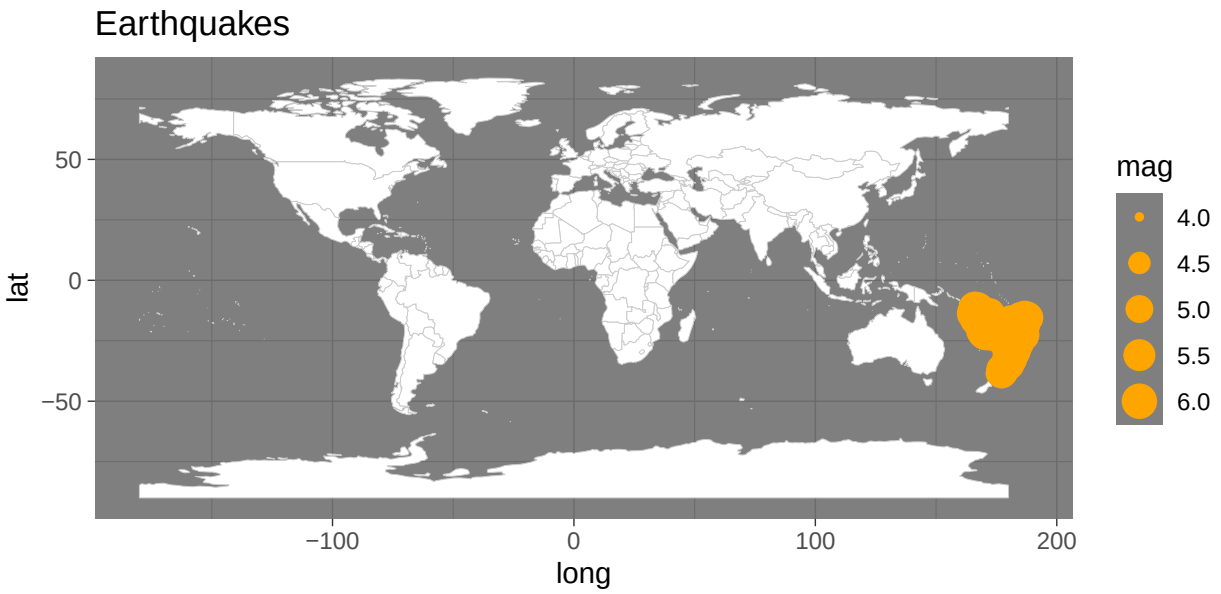


Next, we plot earthquake locations using ggplot on Mercator projection of the earth which is the default method.

```
  p <- ggplot() +
    geom_polygon(data=worldMap, aes(long, lat, group=group), colour='grey80', size=0.1, fill="white") +
    geom_point(data=quakes, aes(x=long, y=lat, size=mag), color="orange") +
    theme_dark() +
    ggtitle("Earthquakes") +
    coord_quickmap()
```

```
## Regions defined for each Polygons
```

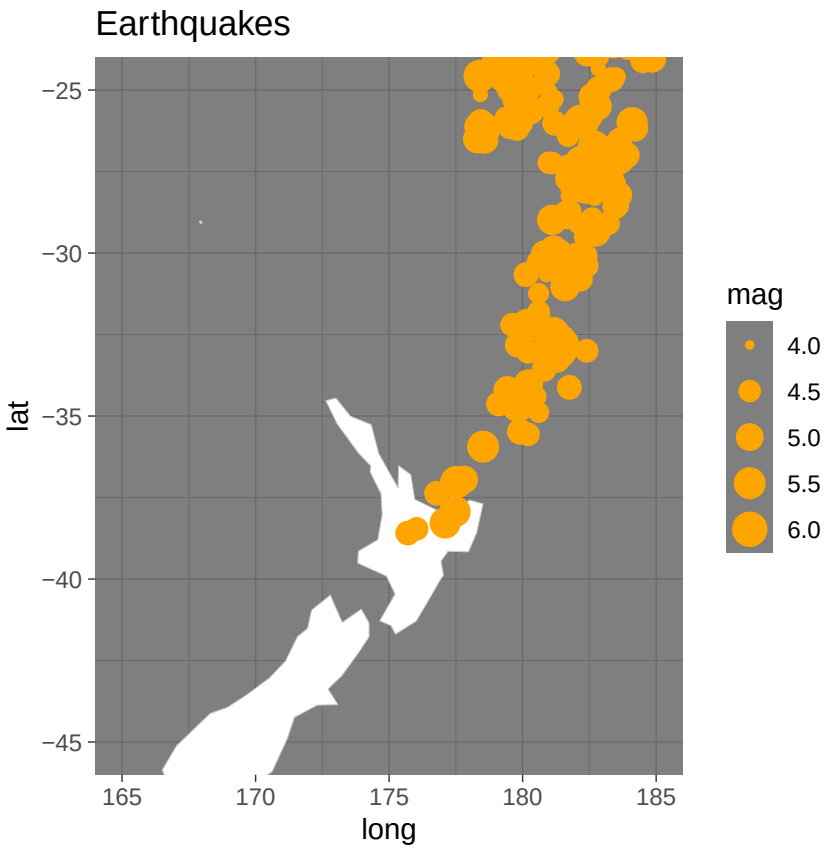
```
  p
```



You can specify limits, e.g., to show epicenters near to New Zealand:

```
  p + coord_quickmap(xlim=c(165,185), ylim=c(-25,-45), expand=TRUE, clip="on")
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```



Detailed maps

More detailed maps are provided by the Center for Spatial Sciences at the University of California, Davis, USA. The data are freely available for academic use and other non-commercial use. Once you have downloaded an appropriate set of spatial polygons you can easily plot the map:

```
gadm <- readRDS("data/gadm36_DEU_2_sp.rds")
plot(gadm, col='lightgrey', border='darkgrey')
```

The visualization of the study centres on spatial data with regional resolution from Germany may look like:

```
ggplot() +
  geom_polygon(data=gadm, aes(long, lat, group=group), colour='grey80', size=0.1, fill="white") +
  geom_point(data=L, aes(x=lng, y=lat, size=size), color="orange") +
  geom_text_repel(data=L, aes(x=lng, y=lat, label=place), color='black', box.padding=unit(0.25, "lines
  ggtitle("Study centre") +
  theme(axis.title.x=element_blank(), axis.title.y=element_blank(), axis.text.x=element_blank(), axis
  coord_map("orthographic")
```

```
## Regions defined for each Polygons
```

Study centre