INFX 573: Problem Set 4 - Data Analysis

Charudatta Deshpande

Due: Thursday, November 2, 2017

Collaborators: Manjiri Kharkar

Instructions:

1. Replace the "Insert Your Name Here" text in the `author:` field with your own full name. Any collaborators must be listed on the top of your assignment.
2. Be sure to include well-documented (e.g. commented) code chucks, figures and clearly written text chunk explanations as necessary. Any figures should be clearly labeled and appropriately referenced within the text.
3. Collaboration on problem sets is fun and useful! However, you must turn in your individual write-up in his or her own words and his or her own work. The names of your collaborators must be listed on each assignment. Do not copy-and-paste from other students' responses or code.
4. When you have completed the assignment and have **checked** that your code both runs in the Console and knits correctly, rename the R Markdown file to `YourLastName_YourFirstName_ps4.Rmd`, knit a PDF and submit both the markdown and the PDF file on Canvas.

The Task

The problem set is inspired by a real-world situation and is deliberately somewhat vague. Your task is to understand the data, convert it into a suitable format, and find the tools that produce the desired output. Note: You are asked to produce a map but you don't have to use dedicated mapping tools like *ggmap* and shapefiles, just ordinary plotting will do.

You are working at PredicitiveAnalytics LLC. One day your Most Important Customer comes to you and says:

I need a temperature and precipitation map of Europe for January and July. It must be based on the most recet NOAA long term means data from NOAA webpage, the v401 format. And I need it by Thursday, November 2nd, 5:30pm. I just need a color map, it does not have to be anything fancy with borders and cities and rivers on it. Just the temperature and rain, plotted in a way I can understand would do.

Download the data and produce such maps for temperature and precipitation (do not use the tools on the website). Make sure to explain and label your data sources and units of measurement. Try to tune the plot with suitable colors, scales, etc, to impress your Important Customer. Comment, or otherwise explain your code, and briefly discuss the results.

Suggestions:

- If you use *ggplot* for plotting, add coordinate transformation `+ coord_map()` (requires *mapproj* library). This ensures the map will be in a valid map projection. You may experiment with different projections.

Solution:

```
# Load Standard libraries
library(ggplot2)
library(mapproj)
```

```
## Loading required package: maps
```

```
library(data.table)
library(tidyverse)
```

```
## Loading tidyverse: tibble
## Loading tidyverse: tidyr
## Loading tidyverse: readr
## Loading tidyverse: purrr
## Loading tidyverse: dplyr
```

```
## Conflicts with tidy packages ----------------------------------------------
```

```
## between():   dplyr, data.table
## filter():    dplyr, stats
## first():     dplyr, data.table
## lag():       dplyr, stats
## last():      dplyr, data.table
## map():       purrr, maps
## transpose(): purrr, data.table
```

```
library(ncdf4)
library(ncdf4.helpers)
library(PCICt)
library(maptools)
```

```
## Loading required package: sp
```

```
## Checking rgeos availability: FALSE
##      Note: when rgeos is not available, polygon geometry     computations in maptools depend on gpcli
##      which has a restricted licence. It is disabled by default;
##      to enable gpclib, type gpclibPermit()
```

```
library(viridis)
```

```
## Loading required package: viridisLite
```

```
library(chron)
library(lattice)
library(RColorBrewer)
library(ggmap)
library(rworldmap)
```

```
## ### Welcome to rworldmap ###
```

```
## For a short introduction type :   vignette('rworldmap')
```

```
library(sp)
#
#####################      PART 1 - Temperature plot    ######################
#
temperature <- nc_open("air.mon.ltm.v401.nc")
lon <- ncvar_get(temperature, "lon")
nlon <- dim(lon)
#head(lon) can be used to view first few records
lat <- ncvar_get(temperature, "lat")
nlat <- dim(lat)
#head(lat) can be used to view first few records
print(c(nlon, nlat))
```

```
## [1] 720 360
```

```
time <- ncvar_get(temperature, "time")
timeunits <- ncatt_get(temperature, "time", "units")
ntime <- dim(time)
#head(time) can be used to view first few records
temp_array <- ncvar_get(temperature, "air")
#
                          # get global attributes
#
title <- ncatt_get(temperature,0,"title")
institution <- ncatt_get(temperature,0,"institution")
datasource <- ncatt_get(temperature,0,"source")
references <- ncatt_get(temperature,0,"references")
history <- ncatt_get(temperature,0,"history")
Conventions <- ncatt_get(temperature,0,"Conventions")
#
#
dlname <- ncatt_get(temperature, "air", "long_name")
dunits <- ncatt_get(temperature, "air", "units")
fillvalue <- ncatt_get(temperature, "air", "_FillValue")
#
#
nc_close(temperature)
#
# split the time units string into fields
tustr <- strsplit(timeunits$value, " ")
tdstr <- strsplit(unlist(tustr)[3], "-")
tmonth = as.integer(unlist(tdstr)[2])
tday = as.integer(unlist(tdstr)[3])
tyear = as.integer(unlist(tdstr)[1])
chron(time, origin = c(tmonth, tday, tyear))
```

```
##  [1] 11/27/23 12/10/25 10/13/27 10/26/29 10/16/31 10/29/33 10/19/35
##  [8] 11/01/37 11/15/39 11/04/41 11/18/43 11/07/45
```

```
#Replace NetCDF fillvalues with R NAs
temp_array[temp_array == fillvalue$value] <- NA
length(na.omit(as.vector(temp_array[, , 1])))
```

```
## [1] 85793
```

```
#Get a single time slice of the data, create an R data frame
m <- 1
temp.slice <- temp_array[, , m]
lonlat <- expand.grid(lon, lat)
temp.vec <- as.vector(temp.slice)
length(temp.vec)
```

```
## [1] 259200
```

```
tmp.dataframe1 <- data.frame(cbind(lonlat, temp.vec))
names(tmp.dataframe1) <- c("lon", "lat", paste("air", as.character(m), sep = "_"))
#head(na.omit(tmp.dataframe1), 20)
#Convert the whole array to a data frame
#
temp.vec.long <- as.vector(temp_array)
length(temp.vec.long)
```

```
## [1] 3110400
```

```
#Then reshape that vector into a 259200 by 12 matrix using the matrix() function,
#and verify its dimensions, which should be 259200 by 12.
temp.mat <- matrix(temp.vec.long, nrow = nlon * nlat, ncol = ntime)
dim(temp.mat)
```

```
## [1] 259200     12
```

```
#
#Create the second data frame from the temp.mat matrix.
lonlat <- expand.grid(lon, lat)
temp.dataframe2 <- data.frame(cbind(lonlat, temp.mat))
names(temp.dataframe2) <- c("lon", "lat", "tmpJan", "tmpFeb", "tmpMar", "tmpApr", "tmpMay",
    "tmpJun", "tmpJul", "tmpAug", "tmpSep", "tmpOct", "tmpNov", "tmpDec")
options(width = 110)
#head(na.omit(temp.dataframe2, 20))
#
#Only keep data for January and July, since that is what we need.
#
temp.dataframe3 <- temp.dataframe2 %>%
                   select(lon, lat, tmpJan, tmpJul)
#Remove all records with value NA.
temp.dataframe4 <- na.omit(temp.dataframe3)
#head(temp.dataframe4)
#Convert temp.dataframe4 into data.table format
as.data.table(temp.dataframe4)
```

```
##           lon    lat    tmpJan      tmpJul
##     1: 316.75  83.25 -34.15334   0.8000001
##     2: 317.25  83.25 -34.25000   0.5933333
##     3: 320.25  83.25 -36.58667  -2.0166667
##     4: 321.25  83.25 -33.70333   0.7766667
##     5: 321.75  83.25 -33.81000   0.7666667
##    ---
## 85789: 357.75 -89.75 -27.63333 -59.5466652
## 85790: 358.25 -89.75 -27.63333 -59.5466652
## 85791: 358.75 -89.75 -27.63333 -59.5466652
## 85792: 359.25 -89.75 -27.63333 -59.5466652
## 85793: 359.75 -89.75 -27.63333 -59.5466652
```

```
#
#Only keep Europe data by calculating Europe limits and creating a new
#data frame with those limits
#
europe.limits <- geocode(c("Cape Fligely, Rudolf Island, Franz Josef Land, Russia", "Gavdos, Greece",
                           "Faja Grande, Azores", "Severny Island, Novaya Zemlya, Russia"))
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Cape%20Fligely,%20Ru
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Gavdos,%20Greece&sens
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Faja%20Grande,%20Azo
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Severny%20Island,%20
```

```
temp.dataframe5 <- subset(temp.dataframe4, lon > min(europe.limits$lon) & lon < max(europe.limits$lon)
                          & lat > min(europe.limits$lat) & lat < max(europe.limits$lat))
#
```

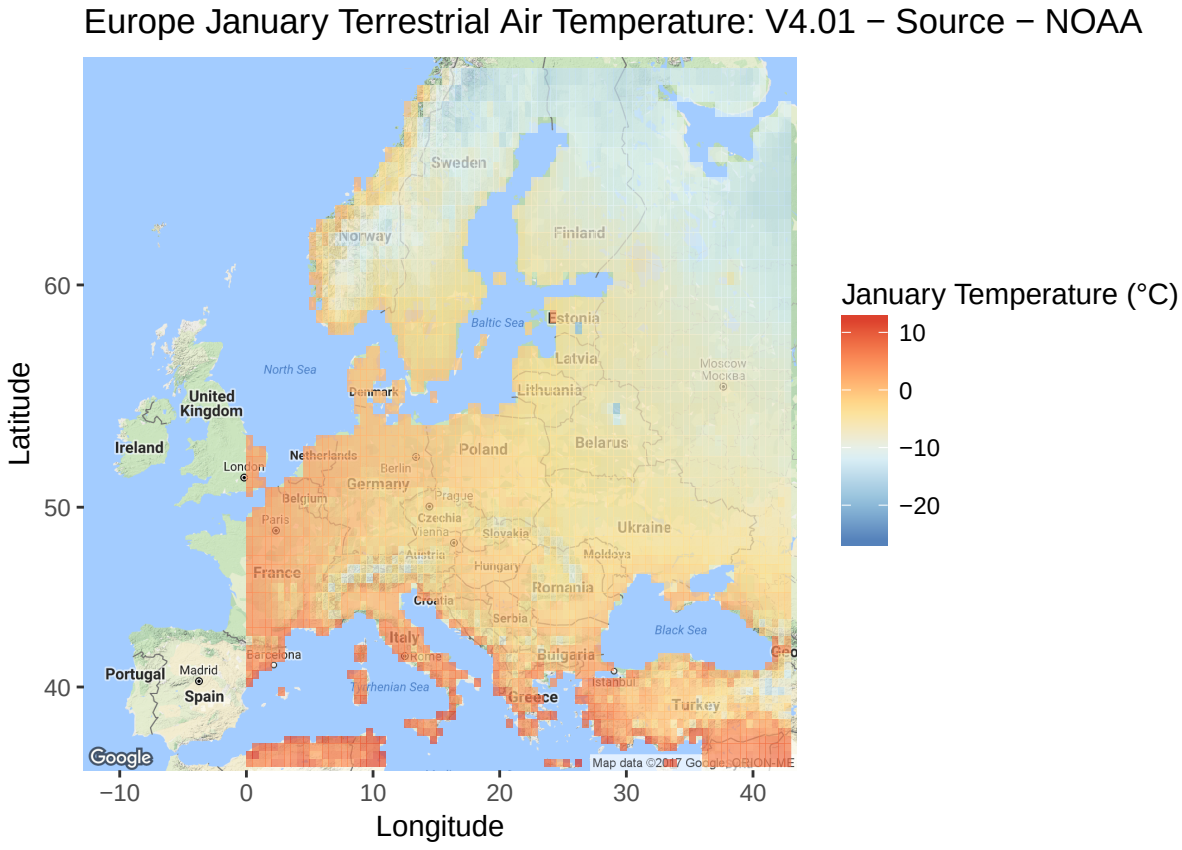
```
#Now we have a dataset with latitude, longitude, January temperature and
#July temperature. Next, we will work on plotting it on Europe map.
#
map <- get_map(location = 'Europe', zoom = 4)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Europe&zoom=4&size=640x640&scale=
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Europe&sensor=false
```

```
#
#Plot 1 - January temperature on Europe map. Use scale_fill_distiller function
#to highlight color.
#
tmpJanMap <- ggmap(map) +
    geom_tile(aes(x = lon, y = lat, fill=temp.dataframe5$tmpJan), data = temp.dataframe5,
              alpha = 0.7) + coord_map() + scale_fill_distiller(palette = "RdYlBu") +
    labs(x="Longitude", y="Latitude", fill = "January Temperature (°C)",
         title = "Europe January Terrestrial Air Temperature: V4.01 - Source - NOAA")
tmpJanMap
```

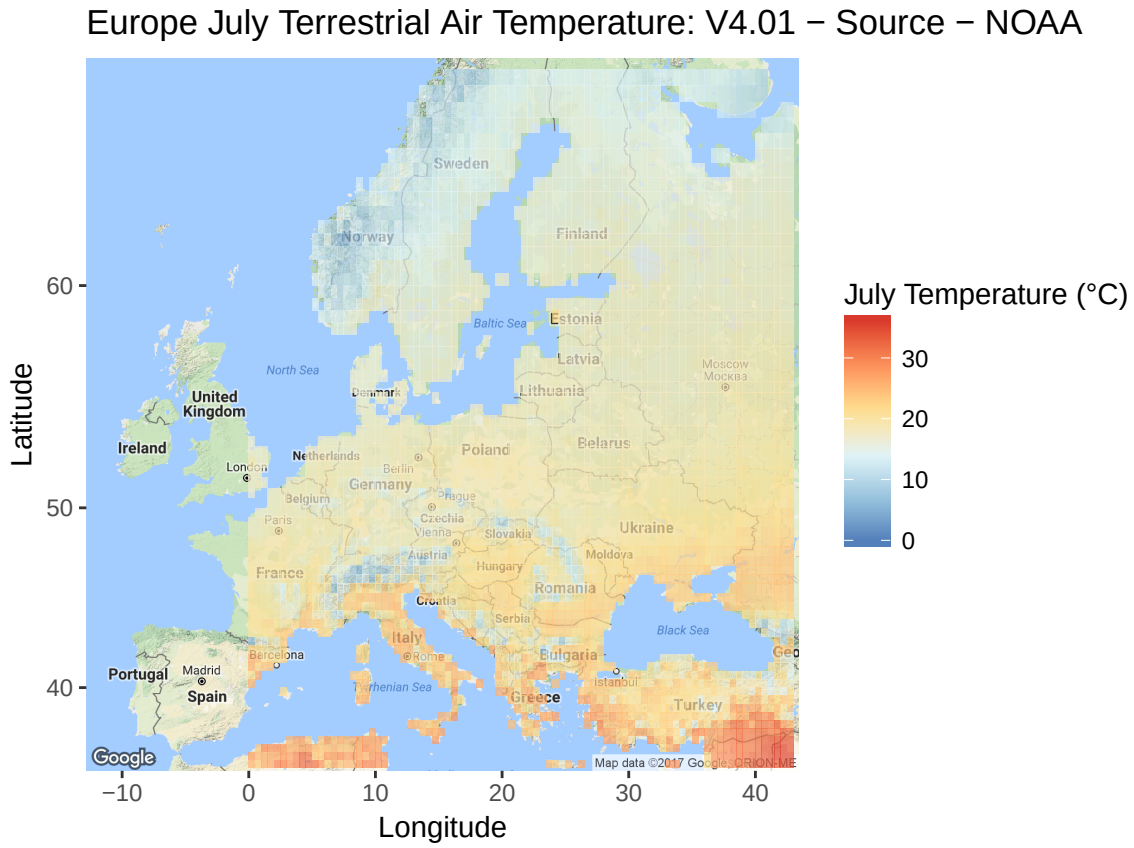
```
## Warning: Removed 2635 rows containing missing values (geom_tile).
```




```
#
#Plot 2 - July temperature on Europe map. Use scale_fill_distiller function
#to highlight color.
#
tmpJulMap <- ggmap(map) +
    geom_tile(aes(x = lon, y = lat, fill=temp.dataframe5$tmpJul), data = temp.dataframe5,
```

```
            alpha = 0.7) + coord_map() + scale_fill_distiller(palette = "RdYlBu") +
    labs(x="Longitude", y="Latitude", fill = "July Temperature (°C)",
         title = "Europe July Terrestrial Air Temperature: V4.01 - Source - NOAA")
tmpJulMap
```

```
## Warning: Removed 2635 rows containing missing values (geom_tile).
```




```
#
#####################      PART 2 - Precipitation plot    ######################
#
precipitation <- nc_open("precip.mon.ltm.v401.nc")
lon <- ncvar_get(precipitation, "lon")
nlon <- dim(lon)
#head(lon) can be used to view first few records
lat <- ncvar_get(precipitation, "lat")
nlat <- dim(lat)
#head(lat) can be used to view first few records
print(c(nlon, nlat))
```

```
## [1] 720 360
```

```
time <- ncvar_get(precipitation, "time")
timeunits <- ncatt_get(precipitation, "time", "units")
ntime <- dim(time)
#head(time) can be used to view first few records
precip_array <- ncvar_get(precipitation, "precip")
#
                          # get global attributes
```

```
#
title <- ncatt_get(precipitation,0,"title")
institution <- ncatt_get(precipitation,0,"institution")
datasource <- ncatt_get(precipitation,0,"source")
references <- ncatt_get(precipitation,0,"references")
history <- ncatt_get(precipitation,0,"history")
Conventions <- ncatt_get(precipitation,0,"Conventions")
#
#
dlname <- ncatt_get(precipitation, "precip", "long_name")
dunits <- ncatt_get(precipitation, "precip", "units")
fillvalue <- ncatt_get(precipitation, "precip", "_FillValue")
#
#
nc_close(precipitation)
#
# split the time units string into fields
tustr <- strsplit(timeunits$value, " ")
tdstr <- strsplit(unlist(tustr)[3], "-")
tmonth = as.integer(unlist(tdstr)[2])
tday = as.integer(unlist(tdstr)[3])
tyear = as.integer(unlist(tdstr)[1])
chron(time, origin = c(tmonth, tday, tyear))
```

```
##  [1] 11/27/23 12/10/25 10/13/27 10/26/29 10/16/31 10/29/33 10/19/35 11/01/37 11/15/39 11/04/41 11/18/
## [12] 11/07/45
```

```
#Replace NetCDF fillvalues with R NAs
precip_array[precip_array == fillvalue$value] <- NA
length(na.omit(as.vector(precip_array[, , 1])))
```

```
## [1] 85598
```

```
#Get a single time slice of the data, create an R data frame
m <- 1
precip.slice <- precip_array[, , m]
lonlat <- expand.grid(lon, lat)
precip.vec <- as.vector(precip.slice)
length(precip.vec)
```

```
## [1] 259200
```

```
precip.dataframe1 <- data.frame(cbind(lonlat, precip.vec))
names(precip.dataframe1) <- c("lon", "lat", paste("precip", as.character(m), sep = "_"))
#head(na.omit(precip.dataframe1), 20)
#Convert the whole array to a data frame
#
precip.vec.long <- as.vector(precip_array)
length(precip.vec.long)
```

```
## [1] 3110400
```

```
#Then reshape that vector into a 259200 by 12 matrix using the matrix() function,
#and verify its dimensions, which should be 259200 by 12.
precip.mat <- matrix(precip.vec.long, nrow = nlon * nlat, ncol = ntime)
dim(precip.mat)
```

```
## [1] 259200     12
```

```
#
#Create the second data frame from the precip.mat matrix.
lonlat <- expand.grid(lon, lat)
precip.dataframe2 <- data.frame(cbind(lonlat, precip.mat))
names(precip.dataframe2) <- c("lon", "lat", "precipJan", "precipFeb", "precipMar", "precipApr",
                              "precipMay", "precipJun", "precipJul", "precipAug", "precipSep",
                              "precipOct", "precipNov", "precipDec")
options(width = 110)
#head(na.omit(precip.dataframe2, 20))
#
#Only keep data for January and July, since that is what we need.
#
precip.dataframe3 <- precip.dataframe2 %>%
                     select(lon, lat, precipJan, precipJul)
#Remove all records with value NA.
precip.dataframe4 <- na.omit(precip.dataframe3)
#head(precip.dataframe4)
#Convert precip.dataframe4 into data.table format
as.data.table(precip.dataframe4)
```

```
##           lon    lat   precipJan    precipJul
##     1: 316.75  83.25 0.891666532 2.4353334904
##     2: 317.25  83.25 0.898333371 2.4356663227
##     3: 320.25  83.25 0.932000041 2.3879997730
##     4: 321.25  83.25 0.951666772 2.3940000534
##     5: 321.75  83.25 0.961666703 2.3886666298
##    ---
## 85423: 357.75 -89.75 0.003666666 0.0006666667
## 85424: 358.25 -89.75 0.003666666 0.0006666667
## 85425: 358.75 -89.75 0.003666666 0.0006666667
## 85426: 359.25 -89.75 0.003666666 0.0006666667
## 85427: 359.75 -89.75 0.003666666 0.0006666667
```

```
#
#Only keep Europe data by calculating Europe limits and creating a new
#data frame with those limits
#
europe.limits <- geocode(c("Cape Fligely, Rudolf Island, Franz Josef Land, Russia", "Gavdos, Greece",
                           "Faja Grande, Azores", "Severny Island, Novaya Zemlya, Russia"))
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Cape%20Fligely,%20Ru
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Gavdos,%20Greece&sens
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Faja%20Grande,%20Azo
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Severny%20Island,%20
```

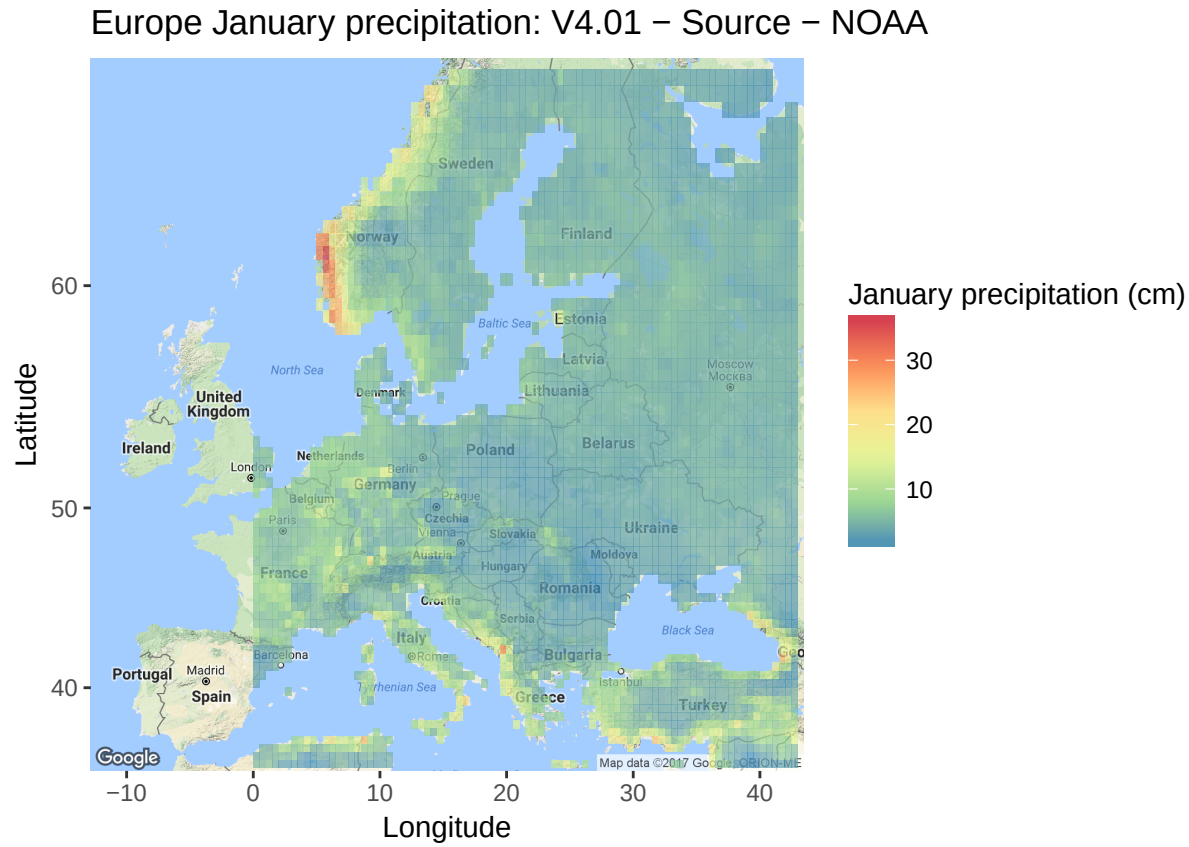
```
precip.dataframe5 <- subset(precip.dataframe4, lon > min(europe.limits$lon) & lon < max(europe.limits$l
                            & lat > min(europe.limits$lat) & lat < max(europe.limits$lat))
#
#Now we have a dataset with latitude, longitude, January precipitation and
#July precipitation. Next, we will work on plotting it on Europe map.
#
map <- get_map(location = 'Europe', zoom = 4)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Europe&zoom=4&size=640x640&scale=
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Europe&sensor=false
```

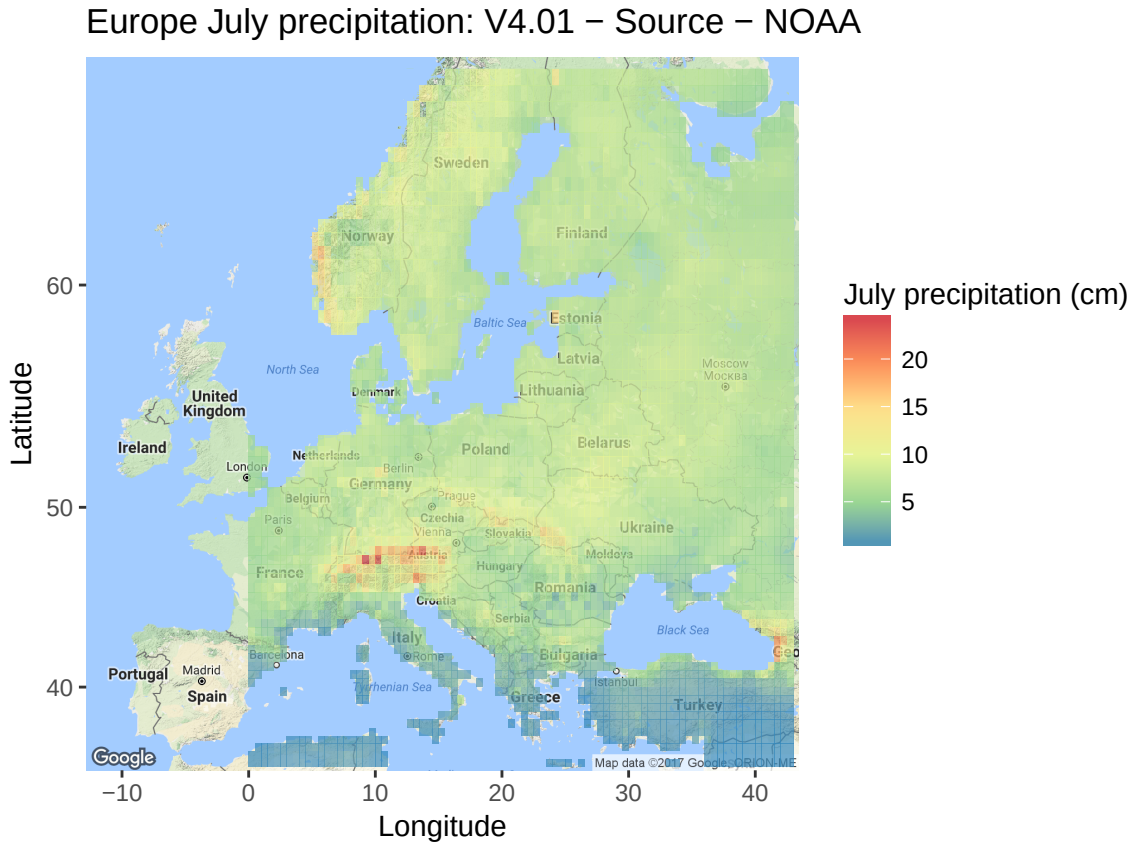
```
#
#Plot 1 - January precipitation on Europe map. Use scale_fill_distiller function
#to highlight color.
#
precipJanMap <- ggmap(map) +
    geom_tile(aes(x = lon, y = lat, fill=precip.dataframe5$precipJan), data = precip.dataframe5,
              alpha = 0.7) + coord_map() + scale_fill_distiller(palette = "Spectral") +
    labs(x="Longitude", y="Latitude", fill = "January precipitation (cm)",
         title = "Europe January precipitation: V4.01 - Source - NOAA")
precipJanMap
```

```
## Warning: Removed 2635 rows containing missing values (geom_tile).
```




```
#
#Plot 2 - July precipitation on Europe map. Use scale_fill_distiller function
#to highlight color.
#
precipJulMap <- ggmap(map) +
    geom_tile(aes(x = lon, y = lat, fill=precip.dataframe5$precipJul), data = precip.dataframe5,
              alpha = 0.7) + coord_map() + scale_fill_distiller(palette = "Spectral") +
    labs(x="Longitude", y="Latitude", fill = "July precipitation (cm)",
         title = "Europe July precipitation: V4.01 - Source - NOAA")
precipJulMap
```

```
## Warning: Removed 2635 rows containing missing values (geom_tile).
```



#

** Results Discussion - **
Source of data - NOAA webpage (https://www.esrl.noaa.gov/psd/data/gridded/data.UDel_AirT_Precip.html)

Units of measurement -
Temperature - Degree Celcius
Precipitation - Centimeter

Europe January Terrestrial Air Temperature -
The temperature varies from -20°C to more than 10°C. As expected, the southern part of Europe is warmer.

Europe July Terrestrial Air Temperature -
The temperature varies from -0°C to almost 40°C. As expected, the southern part of Europe is warmer.

Europe January Precipitation -
Rainfall varies from 0 to almost 40 cm. Almost all parts of Europe get similar rain this month.

Europe July Precipitation -
Rainfall varies from 5 to almost 30 cm. Almost all parts of Europe get similar rain this month.

Concerns -
The main concern I have is that my maps show UK, Ireland, Spain and Portugal as blank, even though they are part of Europe. I tried various ways of making it work but could not. I wanted to analyze deeper but ran out of time. I suspect it could be due to absence of data. I seem to be filtering the data correctly so that does not appear to be the reason.