Making Maps with R—-Intro to Visualizing Spatial Data

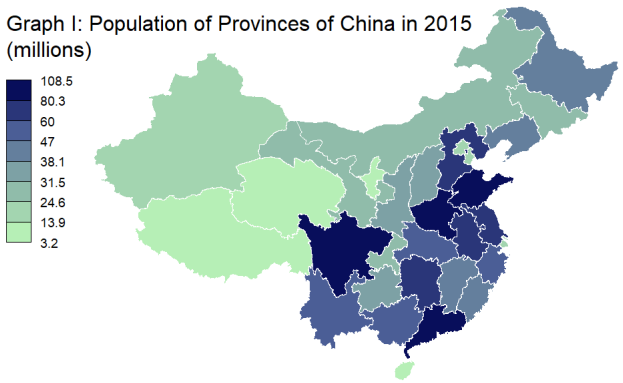
I. INTRODUCTION

One of the most commonly used data analysis approach is drawing comparison between data about various places. For example, analyzing the USA population distribution, the crime concentration within California, and other interregional data is of great value to data scientists.

Now, for example, we want to further analyze the data of distribution of China's population by provinces. If we display these interregional data in a bar chart or a table like the one below, it might not be so clear for people to analyze. The main reason is that we are losing a dimension of information that is the **Spatial information** of those data.

```
##    my_spdf.data.NAME_1 my_spdf.data.pop2015
## 1                Anhui                61.44
## 2              Beijing                21.71
## 3            Chongqing                30.17
## 4               Fujian                38.39
## 5                Gansu                26.00
## 6            Guangdong               108.49
## 7              Guangxi                47.96
## 8              Guizhou                35.30
## 9               Hainan                 9.11
## 10              Hebei                74.25
```

In this case, one way we can retain such information is **to put the data in a map** like the one below.



Isn't it much better this way? Now you can see the pattern of the distribution and probably its relation to the geological features. So in this post I will talk about what data structure do we need to make a map, followed by two ways we can make a data attached map using packages in R.

II. FILE FOR MAPS: INTRO TO SHAPEFILE & .spdf FILE

- We've learnt to store data in .csv or other file forms. But what shall we use to make a map?
  - It's the shapefile that people often use.
- What exactly is a Shapefile?
  - The shapefile format is a digital vector storage format for storing *geometric location* and *associated attribute information*. Simply speaking, it's what format of data we use to make a map with information attached.

```
#Loading a shape file
library(sp)
library(rgdal)

#Store the shapefile as Spdf file in R
#As an example we here load the China's map divided into provinces
my_spdf=readOGR( dsn= getwd() , layer="CHN_adm1")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "C:/Users/Ruiqi/Desktop/MAPS", layer: "CHN_adm1"
## with 31 features
## It has 12 fields
## Integer64 fields read as strings:  ID_0 ID_1 CCN_1
```

- A shapefile is created and often used is GIS. In order to use it in R we first load and convert it into '**.spdf**(Spatial Points Data Frame)' file using package "rgdal"
- spdf file contains mainly two pieces of information:

- Info1: The geometric location is always stored as a list of polygons (e.g. different shapes indicating different provinces)
- Info2: The associated parameters like names,temperatures of the geometric location are always stored as a dataframe of datas matched with the list of polygons (e.g. the population of each province, the temperature of each province)

```
#Info1
summary(my_spdf@polygons)
```

```
##       Length Class    Mode
##  [1,] 1      Polygons S4
##  [2,] 1      Polygons S4
##  [3,] 1      Polygons S4
##  [4,] 1      Polygons S4
##  [5,] 1      Polygons S4
##  [6,] 1      Polygons S4
##  [7,] 1      Polygons S4
##  [8,] 1      Polygons S4
##  [9,] 1      Polygons S4
## [10,] 1      Polygons S4
## [11,] 1      Polygons S4
## [12,] 1      Polygons S4
## [13,] 1      Polygons S4
## [14,] 1      Polygons S4
## [15,] 1      Polygons S4
## [16,] 1      Polygons S4
## [17,] 1      Polygons S4
## [18,] 1      Polygons S4
## [19,] 1      Polygons S4
## [20,] 1      Polygons S4
## [21,] 1      Polygons S4
## [22,] 1      Polygons S4
## [23,] 1      Polygons S4
## [24,] 1      Polygons S4
## [25,] 1      Polygons S4
## [26,] 1      Polygons S4
## [27,] 1      Polygons S4
## [28,] 1      Polygons S4
## [29,] 1      Polygons S4
## [30,] 1      Polygons S4
## [31,] 1      Polygons S4
```

```
#Info2
head(my_spdf@data,2)
```

```
##   ID_0 ISO NAME_0 ID_1  NAME_1 HASC_1 CCN_1 CCA_1       TYPE_1
## 0   49 CHN  China    1   Anhui  CN.AH     0  <NA>       ShÄ<U+009B>ng
## 1   49 CHN  China    2 Beijing  CN.BJ     0  <NA> ZhÃ xiÃ¦shÃ¬
##      ENGTYPE_1     NL_NAME_1 VARNAME_1
## 0     Province å®<U+0089>å¾½|å®<U+0089>å¾½   Ä<U+0080>nhuÄ«
## 1 Municipality å<U+008C><U+0097>äº¬|å<U+008C><U+0097>äº¬ BÄ<U+009B>ijÄ«ng
```

- File Structure The shapefile always comes together with a group of several other files each representing a different aspects of the geodata: .shp - shape format; the feature geometry itself. .shx - shape index format; a positional index of the feature geometry to allow seeking forwards and backwards quickly. .dbf - attribute format; columnar attributes for each shape, in dBase IV format.

- Where to find shape files
- Downloaded Pre-made shape files from different websites(e.g. Geofabrik, OpenMapData.com)
- Customize shapefiles online (e.g. OSM2GIS, BBBike.org)
- Construct it yourself using QGIS or ArcGIS offline

III. Two ways of better analyzing data on a shape file

```
plot(my_spdf,main="China by Provinces")
```

You can already make a plot based on the China sdpf file loaded, but in order to make better analysis and visualization, we can make use of two packages: * Leaflet * Cartography

1.Package “Leaflet”

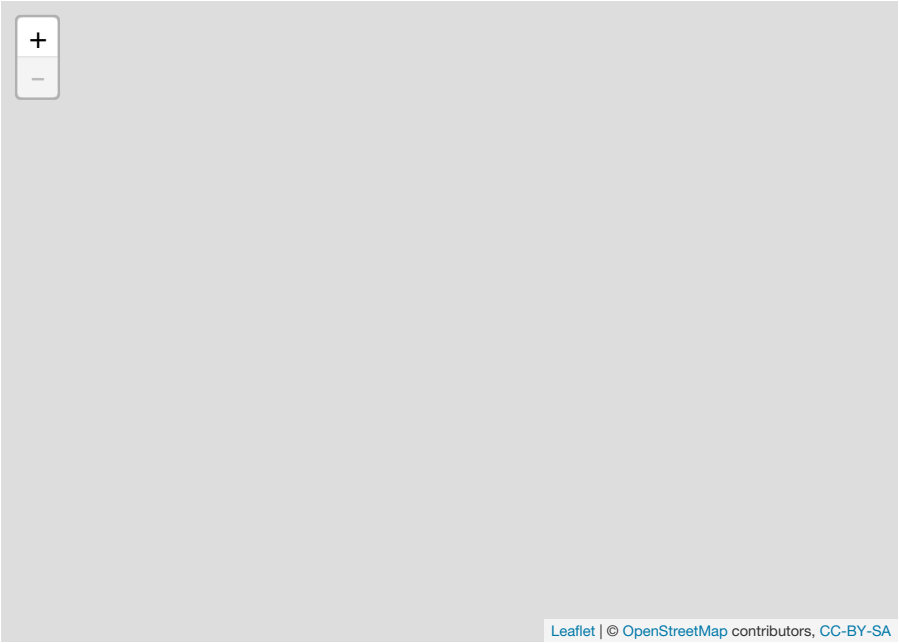
- Lets first take a look on how to use the package “leaflet” to visualize the map and analyze the data attached.
- Leaflet is one of the most popular open-source JavaScript libraries for interactive maps. It’s used by websites ranging from The New York Times and The Washington Post to GitHub and Flickr, as well as GIS specialists like OpenStreetMap, Mapbox, and CartoDB.
- In R instead we can control and integrate the leaflet maps using this package “**Leaflet**”
- Features:
    - Default Base map
    - Various Provider tiles
    - Shorter and simpler Code

Show the basic map

First let’s walk through how we can show the base map of the whole world. This is always the very first thing you wanna do before adding any other layers.

```
#Library
library(leaflet)

# We always initiate a leaflet map with the leaflet() function
m=leaflet()
# Then we Add default OpenStreetMap map tiles(We can add our own map later)
m=addTiles(m)
m
```


```
# Same stuff but using the %>% operator
m=leaflet() %>% addTiles()
m
```



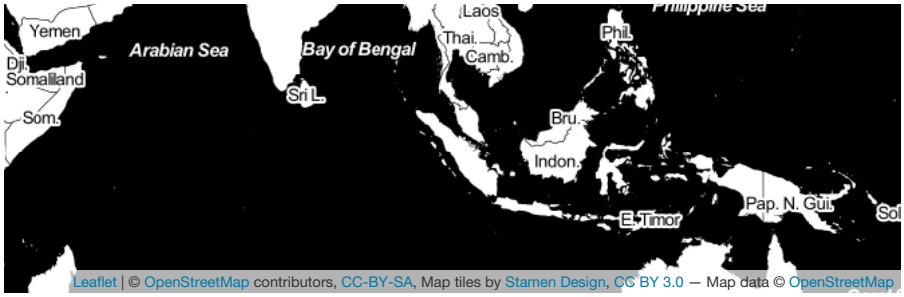
Select Zone(Zoom in & out) Then as we are going to focus on China, we can set our location to China(lat=23,lng=100)

```
# "lng & lat" refer to the longitude & Latitude of the region you want to select, and "zoom" means how large the s
cale is
# For example we want to look at map of Asia
m=leaflet() %>% addTiles() %>% setView( lat=23, lng=100 , zoom=3 )
m
```

Change Background tiles

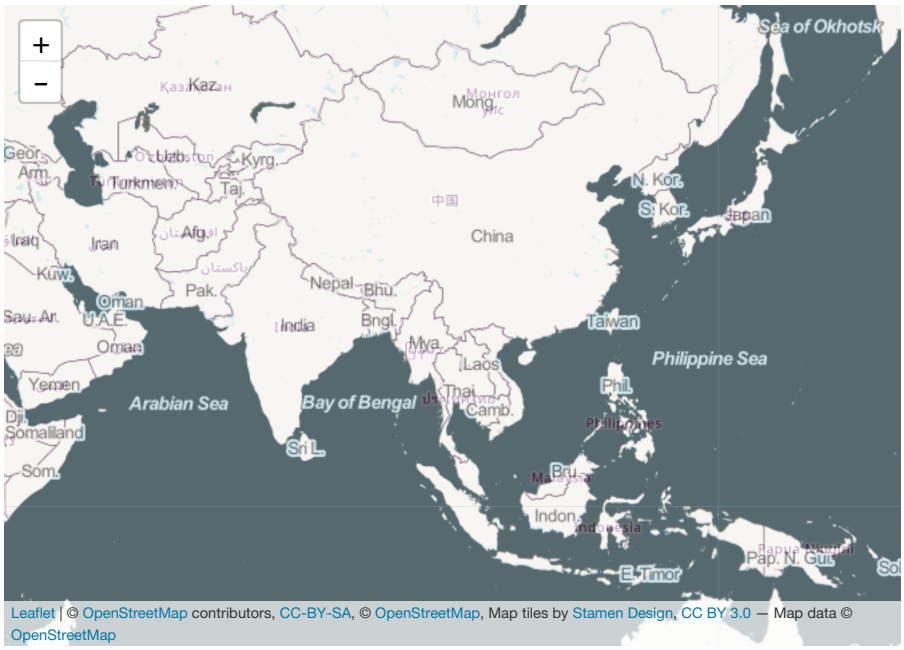
The default background tile, the basemap is the OpenStreetMap, a free editable map of the world. However, you can add a custom tiles to personalize your maps. To choose from the provider's tiles just type providers$ and choose from one of the options. You can also use names(providers) to view all of the options.

```
m=leaflet(my_spdf)%>% addTiles %>% setView( lat=23, lng=100 , zoom=3 )%>% addProviderTiles(providers$Stamen.Toner)
m
```



You can also stack different tiles, but plz make sure you change the opacity of the second layer to make the first visible

```
m=leaflet(my_spdf)%>% addTiles %>% addProviderTiles(providers$OpenStreetMap)%>%addProviderTiles(providers$Stamen.Toner,option=providerTileOptions(opacity=0.5))%>% setView( lat=23, lng=100 , zoom=3 )
m
```



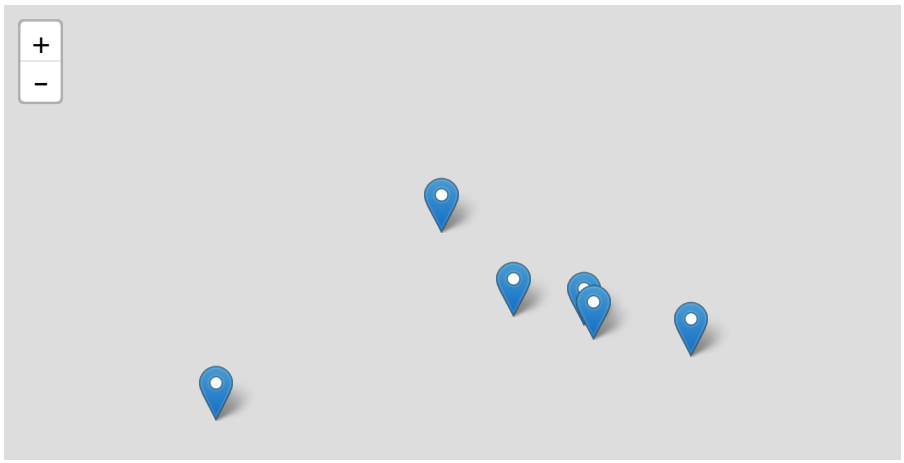
Visualize Area with attached data

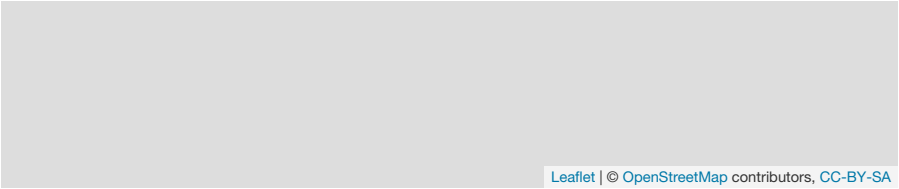
After finishing the set up of the map, now let's begin the visualization process of the data. There mainly are two kinds of data attached to map. * Data collected by point * Data collected by area Let's take a look at each of these.

1. Data collected by point(Add Markers on a Map)

If our data is collected by cities we can show then by adding markers. Here below is how we add markers using leaflet. Just create a dataframe of the longitude and latitude and names of the markers and the corresponding data values and use the addMarkers() in leaflet to put this tile on top.

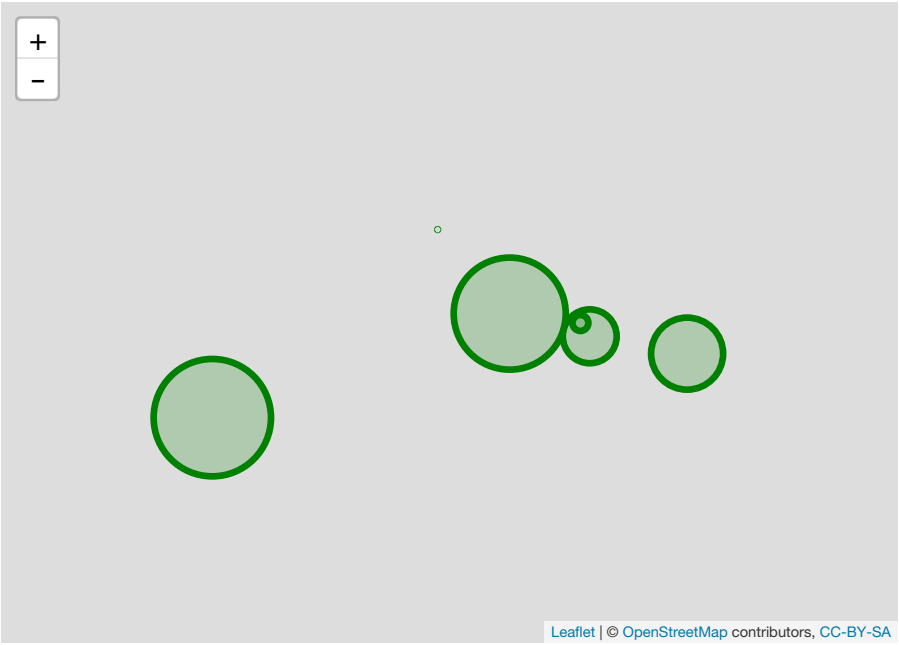
```
# Creating markers of some cities in China
data=data.frame(lng=c(116.38,139.75,126.98,125.75,77.2,106.917) ,  lat=c(39.91,35.68,37.55,39.02,28.6,47.9166) , val=c(21,13.5,10.1,3,22,0.14) , name= c("Beijing","Tokyo","Seoul","Pyongyang","New Delhi","Ulaanbaatar") )

# Show a marker at each position
m=leaflet(data = data) %>% addTiles() %>% addMarkers(~lng, ~lat, popup = ~as.character(name))
m
```





And we can also add circles with size corresponding to the size of data

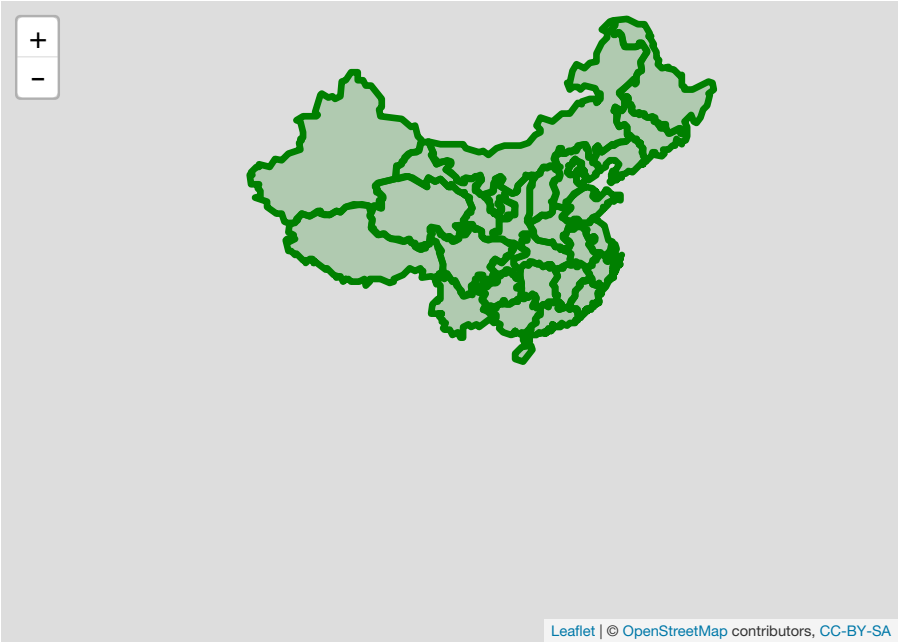
```
# Show a costum circle at each position
m=leaflet(data = data) %>% addTiles() %>%
  addCircleMarkers(~lng, ~lat,
      radius=~val*2 ,
      color="green",
      stroke = TRUE,
      fillOpacity = 0.2,
      popup = ~as.character(name)
      )
m
```



2. Data collected by area (Add costumized tile on leaflet)

- If data is assigned to areas instead of points, we need first to import the boundary tile, and match those polygons with data set.
- To be specific, if we want to hightlight the border of provinces of China, we can add the shapefile 'my_spdf' containing only the border into leaflet. And in this example we attach a color green to it using addPolygons(color="Green").

```
m=leaflet(my_spdf)%>% addTiles()  %>% setView( lat=23, lng=100 , zoom=3) %>%addPolygons(color="Green")
m
```



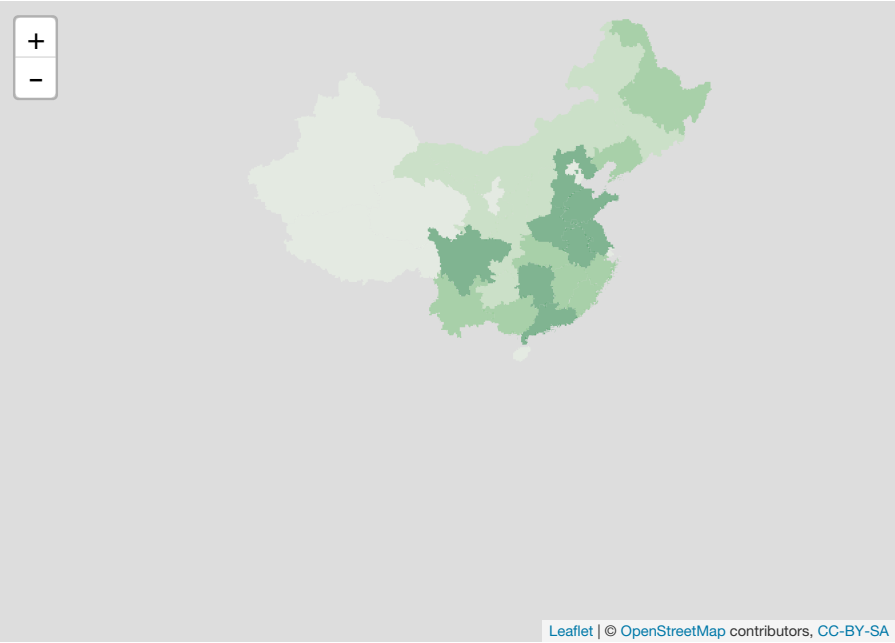
- Once we have the shapes ready, the next step is to attach the corresponding data to each area.
- Right before we dive into how to display those data, first we need to look at how we are going to combine the data we have into the spdf

file so that the data and those polygons can be matched to each other correctly.
    - (To say we are interested in invesgating the population of every provinces in China) We first find the data on internet
    - Then we load the data into a csv file where we write a colunm of NAMEs of the provinces in the same form as those appears in the spdf@data file, otherwise when we combine the two data frames those data cannot be matched with the correct area.

```
CHN <- read.csv("CHNLocations2.csv",header = TRUE)
DF2 <- merge.data.frame(CHN,my_spdf@data)
my_spdf@data <- DF2
```

Displaying the population variable & attach it with parallel color degree

```
m=leaflet(my_spdf)%>% addTiles()  %>% setView( lat=23, lng=100 , zoom=3) %>%
  addPolygons( stroke = FALSE, fillOpacity = 0.5, smoothFactor = 0.5, color = ~colorQuantile("Greens", pop2015)(po
p2015) )
m
```

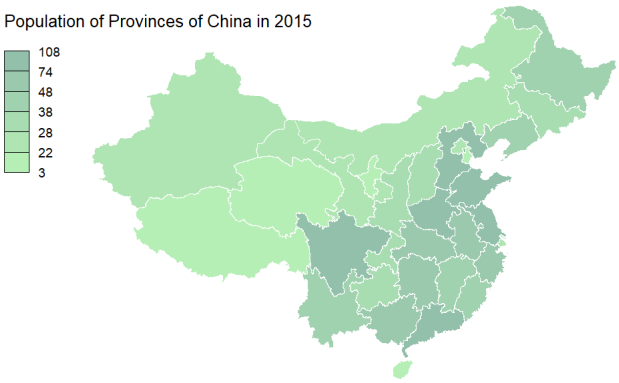


2.Package Cartography

- We can do the same choropleth map using another package called Cartography.
- Features:
    - No base map
    - More display methods(e.g. bars of variables on map)

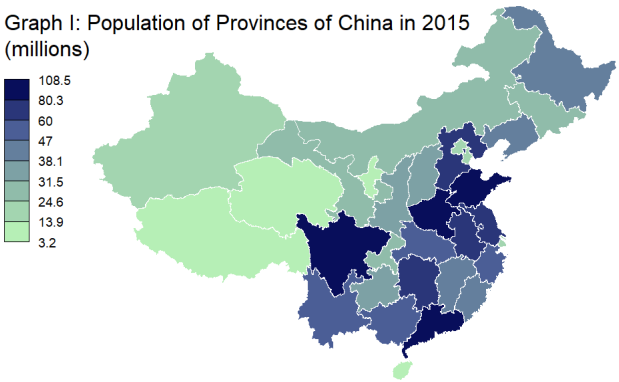
Construct the map using “choroLayer”

```
#Loading the package
library(cartography)

#'spdf' is the map file we want to show
#'df' is the data file inside the spdf file
#'var' is the variable we want to compare across regions
choroLayer(spdf = my_spdf,
           df = my_spdf@data,
           var = "pop2015",
           border = "white",
           lwd = 0.4,
           col = carto.pal(pal1 = "turquoise.pal", n1 = 20),
           legend.pos = "topleft",
          legend.title.txt = 'Population of Provinces of China in 2015')
```



Below here is the map I showed at the beginning of the post

```
choroLayer(spdf = my_spdf,
           df = my_spdf@data,
           var = "pop2015",
           method = "quantile",
           nclass = 8,
           border = "white",
           lwd = 0.4,
           col = carto.pal(pal1 = "turquoise.pal", n1 = 8),
           legend.pos = "topleft",
           legend.title.txt = 'Graph I: Population of Provinces of China in 2015\n(millions)',
           legend.title.cex = 1,
           legend.values.rnd = 1)
```

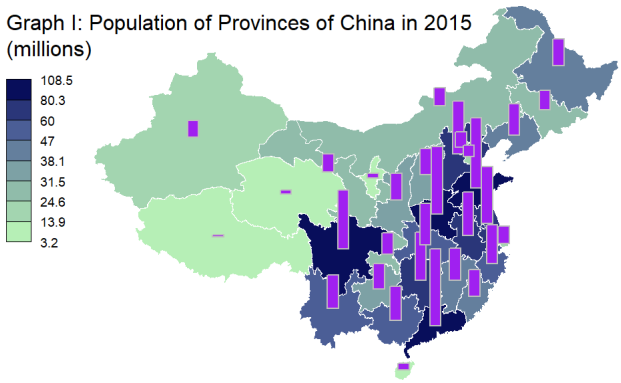


Ways of displaying data other than color quantile

package cartography offers many convinient ways of display data, let's take a look at how can we display the population of China's provinces in bars on the map.

```
choroLayer(spdf = my_spdf,
           df = my_spdf@data,
           var = "pop2015",
           method = "quantile",
           nclass = 8,
           border = "white",
           lwd = 0.4,
           col = carto.pal(pal1 = "turquoise.pal", n1 = 8),
           legend.pos = "topleft",
           legend.title.txt = 'Graph I: Population of Provinces of China in 2015\n(millions)',
           legend.title.cex = 1,
           legend.values.rnd = 1)
propSymbolsLayer(spdf = my_spdf,
                 df = my_spdf@data,
                 var = "pop2015",
                 symbols = "bar",
                 inches = 0.6,
                 col = "purple",
                 legend.pos = "n",
                 border = "grey")
```

```
## Warning in st_centroid.sfc(st_geometry(x), of_largest_polygon =
## of_largest_polygon): st_centroid does not give correct centroids for
## longitude/latitude data
```



IV. **CONCLUSION**

So far we have gained the basic knowledge about the shape file and spdf file that make up a map in R. And we have learnt how to utilize the two efficient packages “leaflet” and “cartography” and what each of them is good at. Next time when there is a data set containing information from various places I hope you could remember to use what you learnt above to make a nice looking map.

Take Home Messages

- Shape file(.shp) is how maps are stored
- .spdf file is how maps are stored in R
- package leaflet functions
    - leaflet()
    - addTiles()
    - setView(lat, lng , zoom)
    - addPolygons(color = ~colorQuantile(“”, var)(var))
- package cartography
    - choroLayer(spdf,df,var,col = carto.pal(pal1 = “turquoise.pal”, n1 = 8),
    - propSymbolsLayer(spdf,df,va,symbols)

V. **REFERENCE**

https://en.wikipedia.org/wiki/Shapefile

https://mgimond.github.io/Spatial/introGIS.html

http://wiki.openstreetmap.org/wiki/Shapefiles

http://www.r-graph-gallery.com/portfolio/maps/

https://www.openstreetmap.org/about

https://rstudio.github.io/leaflet/markers.html
https://www.statista.com/statistics/279013/population-in-china-by-region/
https://rstudio.github.io/leaflet/