post01

Jason Liu

2017/10/26

Post 01 - Drawing Maps Using `ggplot2`, `maps`, and `mapdata`

1. Introduction

In this post, I will introduce the package `mapdata` which contains the map data of China, Japan, New Zealand. I will also introduce the package `maps` which contains the world map and other various maps. With the detailed data in these packages, we can draw the maps of these countries and even the whole world easily using the package `ggplot2`. It is useful to learn how to draw maps using R because maps can help us learn the world and analyze data.

In addition, I will transform the maps from Cartesian Coordinates to Polar Coordinates. To do so, I will introduce the syntax `coord_map`.

2. Maps of Countries

Now, let's start.

Firstly, we will load the packages `ggplot2`, `mapdata`, and `maps`. `ggplot2` is a package that we are familiar with. What is `mapdata` and `maps`? `mapdata` and `maps` are packages containing detailed and rich data of several countries and even the whole world. They are data packages rather than syntax or code packages.

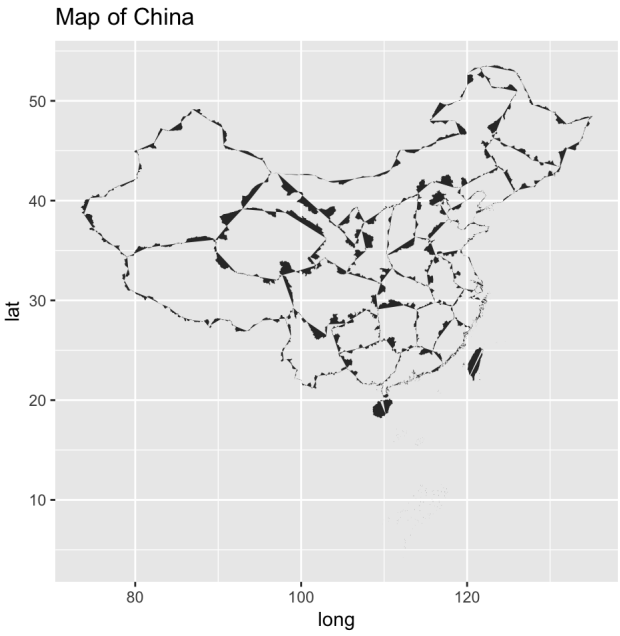
```
# Load packages `ggplot2` and `mapdata`
library(ggplot2)
library(mapdata)
```

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

```
library(maps)
```

Next, let's draw the map of China.

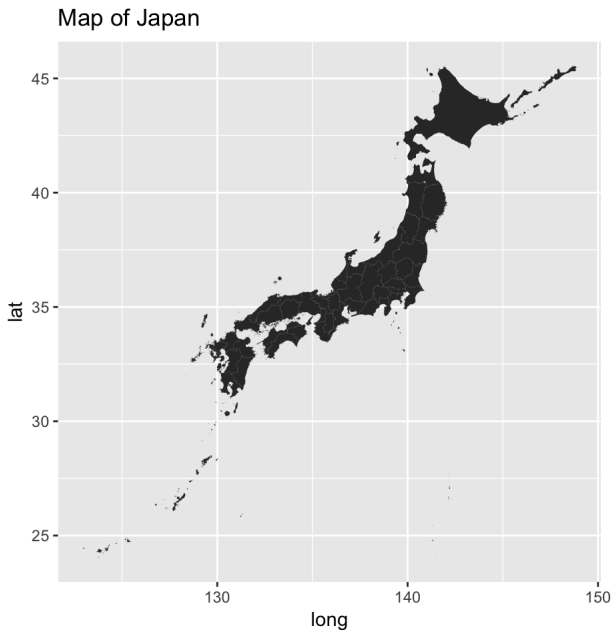
```
# Define China as the data 'china' in the package `mapdata`, and use the code `map_data`
China <- map_data('china')
# Use `ggplot` to draw the map. 'Long', 'lat', and 'group' are three variables in 'China'.
ggplot() +
  # Use `geom_polygon` to draw maps
  geom_polygon(data = China, aes(x = long, y = lat), group = China$group) +
  # `coord_fixed` stretch or shrink the map, and 1.2 is a good value.
  coord_fixed(1.2) +
  # add a title to this map
  ggtitle('Map of China')
```



This is a map of China and it clearly shows the borders of different provinces in it.

Now, let's draw the map of Japan.

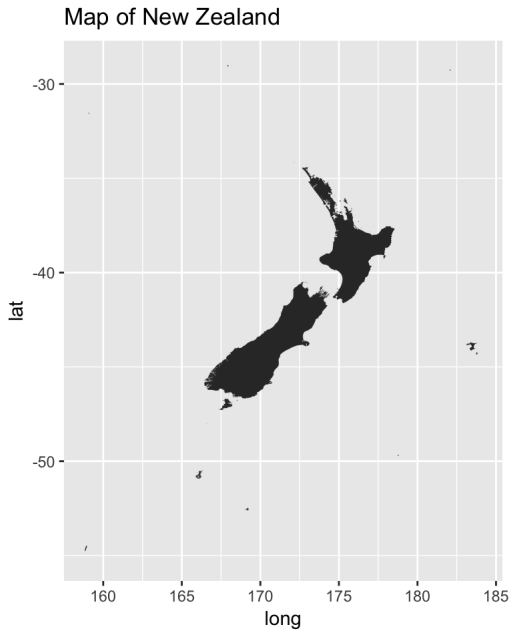
```
# Define Japan as the data 'japan' in the package `mapdata`, and use the code `map_data`
Japan <- map_data('japan')
# Use `ggplot` to draw the map. 'Long', 'lat', and 'group' are three variables in 'Japan'.
ggplot() +
  geom_polygon(data = Japan, aes(x = long, y = lat), group = Japan$group) +
  coord_fixed(1.2) +
  ggtitle('Map of Japan')
```



This is a map of Japan and it clearly shows the borders of different prefectures in it.

Now, let's draw the map of New Zealand.

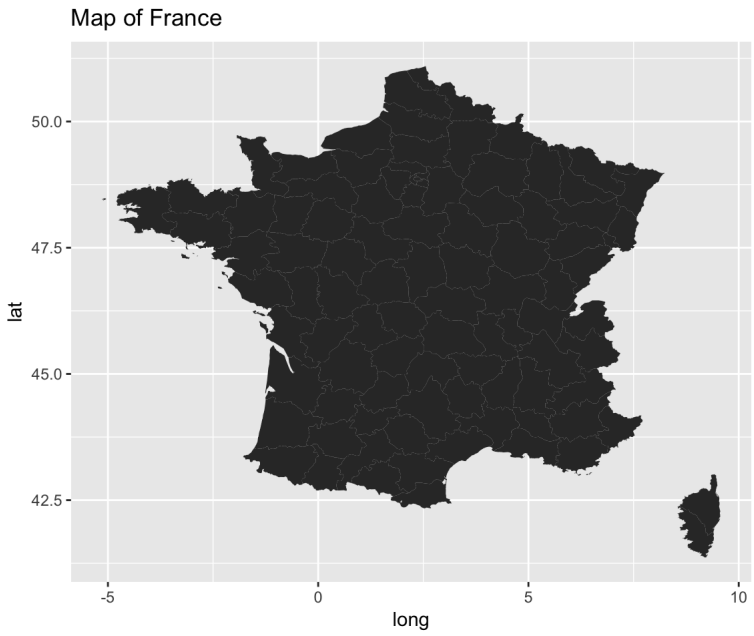
```
# Define New_Zealand as the data 'nzHires' in the package `mapdata`, and use the code `map_data`
New_Zealand <- map_data('nzHires')
# Use `ggplot` to draw the map. 'Long', 'lat', and 'group' are three variables in 'nzHires'.
ggplot() +
  geom_polygon(data = New_Zealand, aes(x = long, y = lat), group = New_Zealand$group) +
  coord_fixed(1.2) +
  ggtitle('Map of New Zealand')
```



This is a map of New Zealand.

Now, let's draw the map of France.

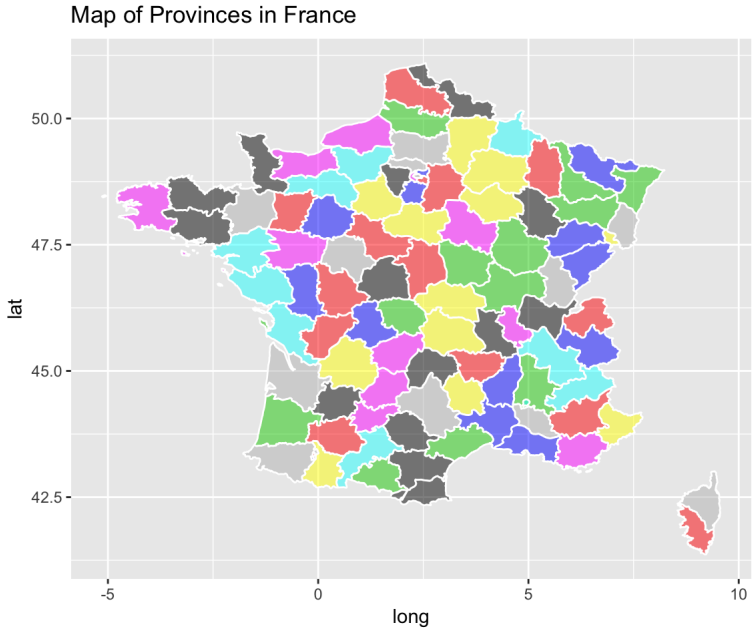
```
# Define France as the data 'france' in the package `maps`, and use the code `map_data`
France <- map_data('france')
# Use `ggplot` to draw the map. 'Long', 'lat', and 'group' are three variables in 'france'.
ggplot() +
  geom_polygon(data = France, aes(x = long, y = lat), group = France$group) +
  coord_fixed(1.2) +
  ggtitle('Map of France')
```



This is a map of France and it clearly shows the borders of different provinces in it.

We can even go a step further. For the data in `maps`, we can give different provinces different colors.

```
France <- map_data('france')
ggplot() +
  # Use `fill` to give different provinces different colors
  # 'group' is a variable in 'France', and different numbers in 'group' represent different provinces
  geom_polygon(data = France, aes(x = long, y = lat), group = France$group, fill = France$group,
               # `color` = white changes the border of provinces into white, and `alpha` changes the
transparency of this map
               color = 'white', alpha = 0.5) +
  coord_fixed(1.2) +
  ggtitle('Map of Provinces in France') +
  guides(fill=FALSE)
```

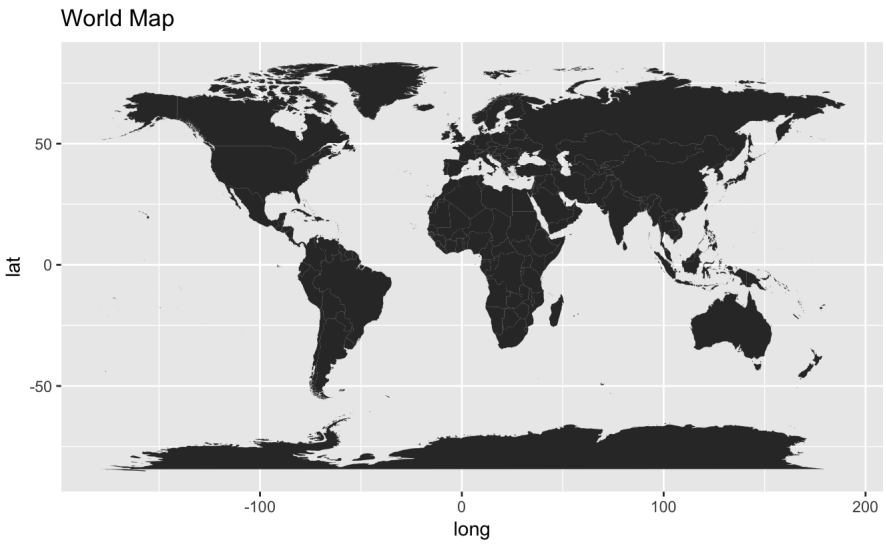


This is the map of France which shows different provinces in France.

Now, let's draw the world map!

3. World Map

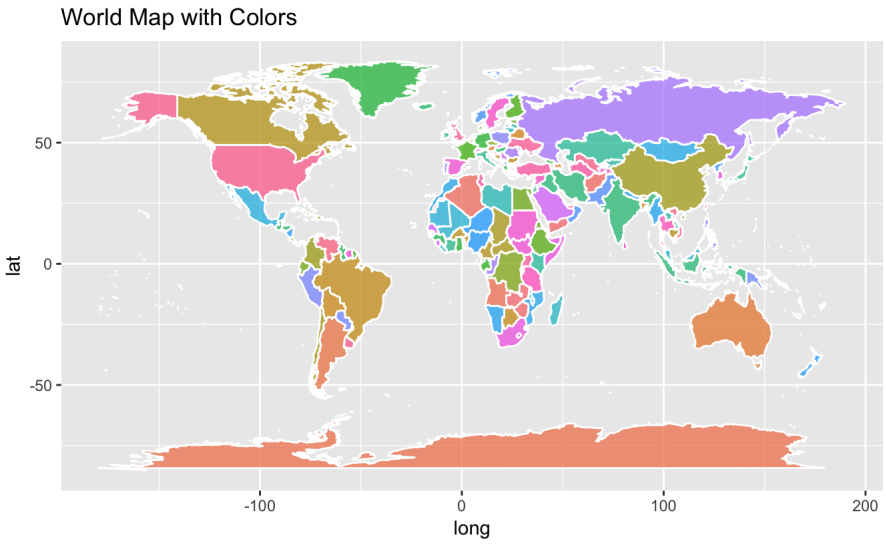
```
# Define World as the data 'world' in the package `maps`, and use the code `map_data`
World <- map_data('world')
# Use `ggplot` to draw the map. 'Long', 'lat', and 'group' are three variables in 'world'.
ggplot() +
  geom_polygon(data = World, aes(x = long, y = lat), group = World$group) +
  coord_fixed(1.2) +
  ggtitle('World Map')
```



This is the world map! It shows the border of different countries clearly.

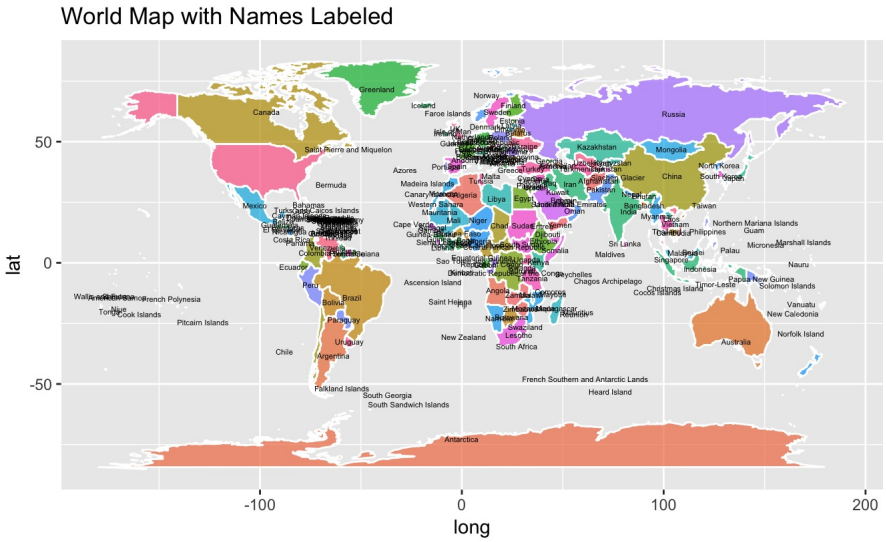
Let's give different colors to different countries using the knowledge we just learned.

```
ggplot() +
  # Use `fill` to give different countries different colors
  # 'region' is a variable in 'world', and different names in 'region' represent different countries
  geom_polygon(data = World, aes(x = long, y = lat, fill = region), group = World$group, color = 'white', alpha =
0.7) +
  coord_fixed(1.2) +
  ggtitle('World Map with Colors') +
  guides(fill=FALSE)
```



This is the world map with colors. Now, let's show the names of those countries and areas on the map!

```
# define 'center' as the center a country or an area
center <- aggregate(cbind(long, lat) ~ region, data = World,
                    FUN=function(x) mean(range(x)))
# Graph it
ggplot() +
  geom_polygon(data = World, aes(x = long, y = lat, fill = region), group = World$group, color = 'white', alpha =
0.7) +
  coord_fixed(1.2) +
  ggtitle('World Map with Names Labeled') +
  guides(fill=FALSE) +
  # add labels to the map
  geom_text(data = center, aes(long, lat, label = region), size = 1.5)
```

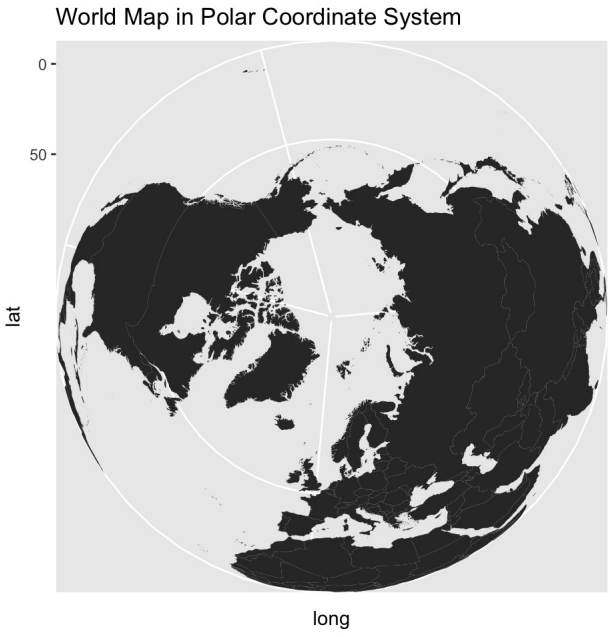


This is the world map with names. As you can see, the names of some countries and areas are seemingly not at the center of their territories. That's because these countries also have oversea territories and the names of these countries must be at the center of their overall territories.

4. Maps in Polar Coordinate System using `coord_map`

Now, let's transform the world map from Cartesian Coordinates to Polar Coordinates.

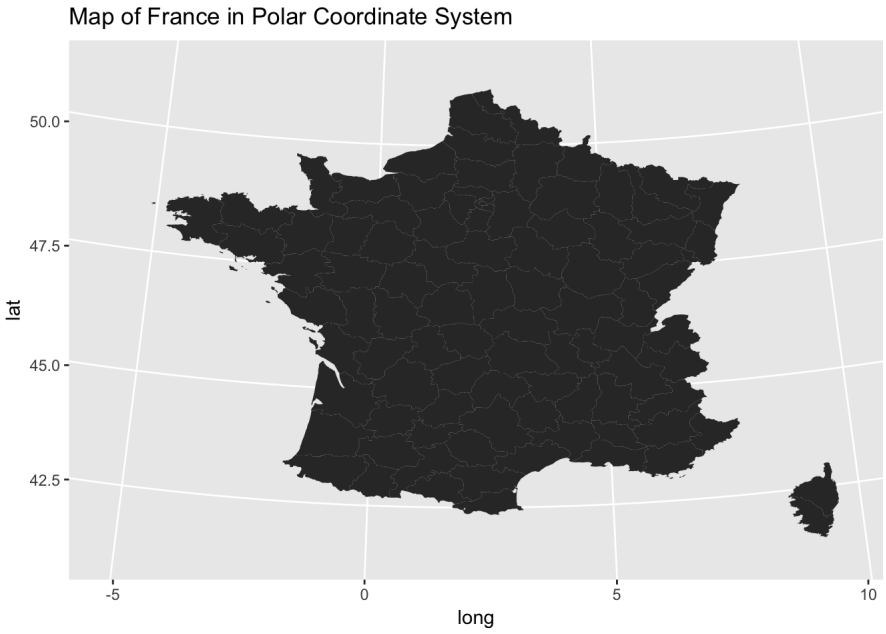
```
# Define World_Map at first so that we can call it easily later
World_Map <- ggplot() +
  geom_polygon(data = World, aes(x = long, y = lat), group = World$group)

# Using `coord_map` to transform the world map from Cartesian Coordinates to Polar Coordinates
World_Map + coord_map(projection = 'ortho') +
  # Add a new title to this map
  ggtitle('World Map in Polar Coordinate System')
```



This is the world map in polar coords! We can see how the world look like more intuitively.

Now, let's transform the Map of France from Cartesian Coords to Polar Coords using the same method.

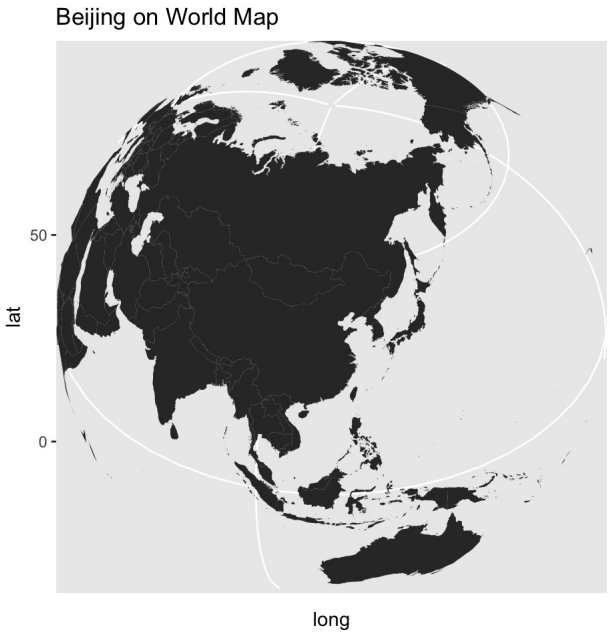
```
# Define World_Map at first so that we can call it easily later
France_Map <- ggplot() +
  geom_polygon(data = France, aes(x = long, y = lat), group = France$group)

# Using `coord_map` to transform the map of France from Cartesian Coordinates to Polar Coordinates
France_Map + coord_map(projection = 'ortho') +
  # Add a new title to this map
  ggtitle('Map of France in Polar Coordinate System')
```



This is the map of France in Polar Coords.

Now, let's locate a certain city on the World Map in Polar Coords. Let's locate Beijing's position (40N, 116E).

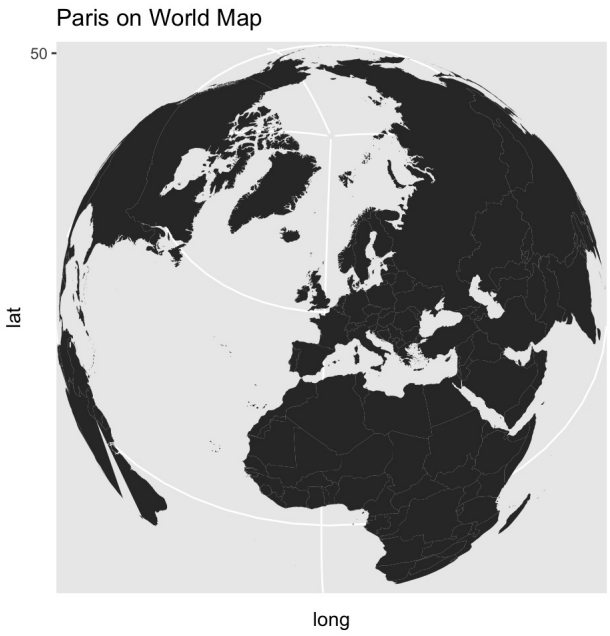
```
World_Map +
  # `orientation` should be the longitude and latitude of a position.
  # `projection` = 'ortho' means to look down from North Pole.
  coord_map(projection = 'ortho', orientation = c(40, 116, 0)) +
  ggtitle('Beijing on World Map')
```



The center of this map is Beijing!

Now, let's locate the position of Paris (49N, 2E) on the World Map.

```
World_Map +
  # A new orientation.
  coord_map(projection = 'ortho', orientation = c(49, 2, 0)) +
  ggtitle('Paris on World Map')
```



The center of this map is where Paris at.

5. Conclusion

This post tells you how to graph various maps using the package `ggplot2` and two data packages `maps` and `mapdata`. We can use `geom_polygon` to draw a map and use `fill` to give different colors to different provinces or countries on the map. In addition, we can use `coord_map` to transform a map from Cartesian coords to Polar Coords. And We can use `orientation` to locate a city on a world map. After you read the post, you will be able to graph a beautiful map using the three packages and the syntax above.

6. Reference

1. http://ggplot2.tidyverse.org/reference/geom_text.html I used this resource to refresh my memory about `geom_text` and how to label on a graph.
2. http://eriqande.github.io/rep-res-web/lectures/making-maps-with-R.html This resource gave me some insight into how to draw a basic map using `ggplot2`, and I learned the syntax `geom_polygon` from it.
3. http://ggplot2.tidyverse.org/reference/coord_map.html From this resource, I learned how to use `coord_map`.
4. http://ggplot2.tidyverse.org/reference/geom_polygon.html From this resource, I learned `geom_polygon` further and knew more about it.
5. https://cran.r-project.org/web/packages/maps/maps.pdf From this resource, I knew what is included in the `maps` package.
6. https://cran.r-project.org/web/packages/mapdata/mapdata.pdf From this resource, I knew what is included in the `mapdata` package.
7. https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf It's the first time I knew that I can use `ggplot2` to graph a map.