The maps package: a new way of making maps

Wenqu Wang

Introduction

During lab12, we took a first look at mapping objects in R using the `"ggmap"` and `"RgoogleMaps"` package. For example, we want to plot the map of San Francisco using `"RgoogleMaps"`.

The first step is to download the packages required and read the required data.

```
setwd('/Users/apple/Desktop/fall 2017/Stat 133/stat133-hws-fall17/post02')
#download packages
require("RgoogleMaps")
```

```
## Loading required package: RgoogleMaps
```

```
library(RgoogleMaps)
#read data (the data we created during lab11)
dat <- read.csv(file = "/Users/apple/Desktop/fall 2017/Stat 133/stat133-hws-fall17/post02/data/maps.csv")
```

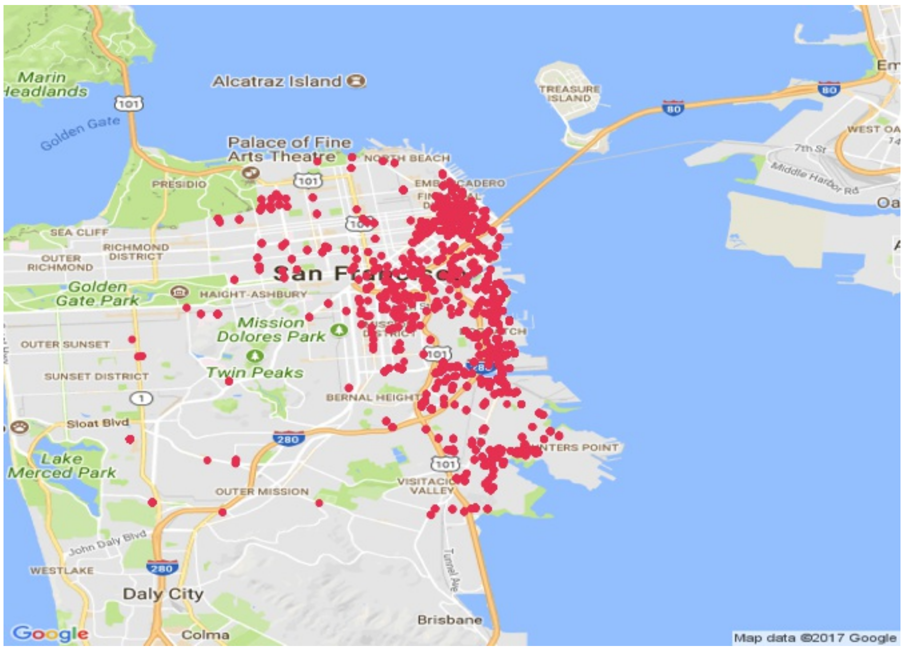
Using the function `GetMap()` which requires a `center` and a `zoom` specifications to get a map of SF.

```
# coordinates for center of the map
center <- c(mean(dat$lat, na.rm = TRUE), mean(dat$lon, na.rm = TRUE))

# zoom value
zoom <- min(MaxZoom(range(dat$lat, na.rm = TRUE),
                    range(dat$lon, na.rm = TRUE)))

# san francisco map
map1 <- GetMap(center=center, zoom=zoom, destfile = "/Users/apple/Desktop/fall 2017/Stat 133/stat133-hws-fa
```

```
PlotOnStaticMap(map1, dat$lat, dat$lon, col = "#ed4964", pch=20)
```



As you can see in this long example, making map using `"RgoogleMaps"` requires a lot of effort. First you need to import a data frame containing the data you need, like longitude and latitude. Even if there's a lot of existing data online, you'll need a lot of time to clean the data to optimize your need.

Actually, a better and simpler way of making maps does exist, that is using the `"maps"` package, which contains mapping data and `"ggplot2"`, which is a tool that we are very familiar with at this point.

Goals

- Introduce the readers about making maps using `"maps"` package.
- Introduce the readers how to convert the data in `"maps"` and `"mapdata"` into data that ggplot2 could deal with.

What is `"maps"` and `"mapdata"` ?

`"maps"` is a package that contains data we need for plotting maps and `"mapdata"` is a supplement that contains more data. In `"maps"` and `"mapdata"` we could find mapping data for different countries.

Map of the United States

Now let's get s tarted. Suppose we want to make a map for the US.

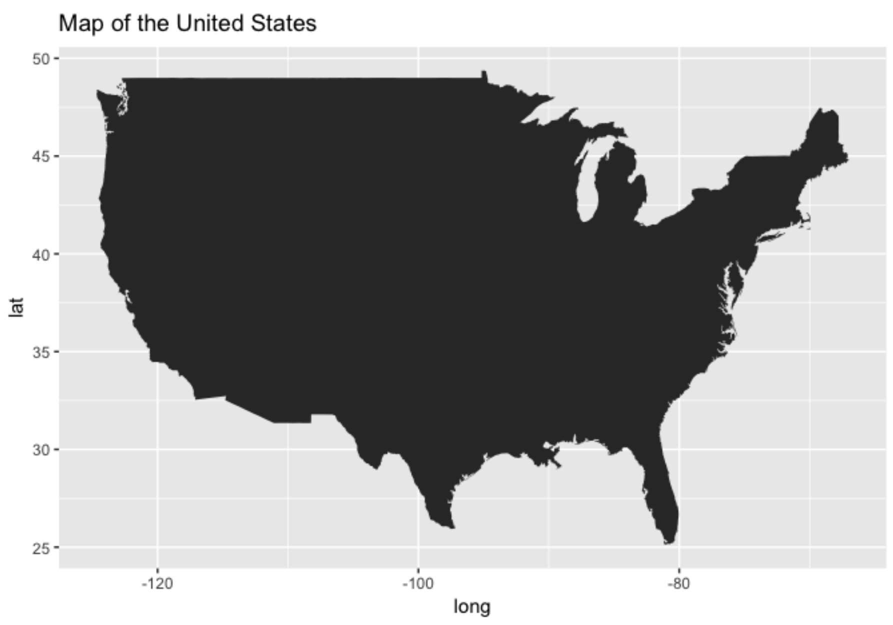
The first step is always to load the required packages.

```
#loading all required packages used in this post
library(maps)
library(mapdata)
library(ggplot2)
```

To fetch the data we need when ploting the USA map, we use a function called `"map_data"` , which returns a data frame containing the specific data set we need.

```
#extract usa from "maps" using map_data
usa <- map_data("usa")
```

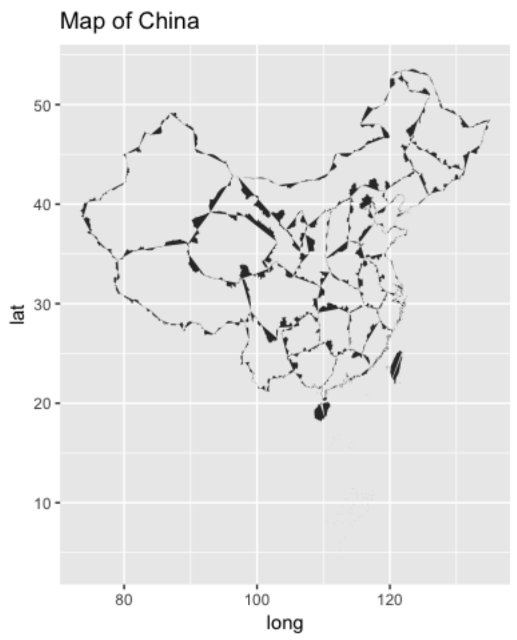
```
# a simple black map of USA
ggplot() + geom_polygon(data = usa, aes(x=long, y = lat), group = usa$group) +
  coord_fixed(1.5) + ggtitle("Map of the United States")
```



In the same way, we could plot the map of China.

```
#extract usa from "mapdata" using map_data
China <- map_data("china")
```

```
# a simple black map of China
ggplot() + geom_polygon(data = China, aes(x=long, y = lat), group = China$group) +
  coord_fixed(1.5) + ggtitle("Map of China")
```



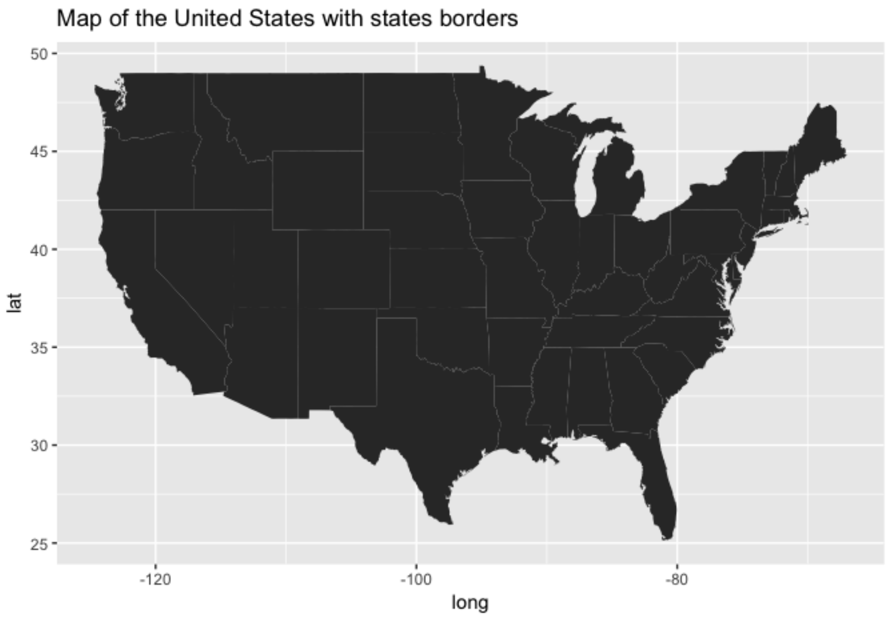
You may noticed that the output of the two maps seems different although we used exactly the same code. This is because the data we extract are different. The "usa" data does not contain borders between each states which the "china" data set does.

So, what if we want to show the borders between each states?

We have another data set called `"state"`, which contains data about each state of the United States.

```
#extract usa from "maps" using map_data
states <- map_data("state")
```

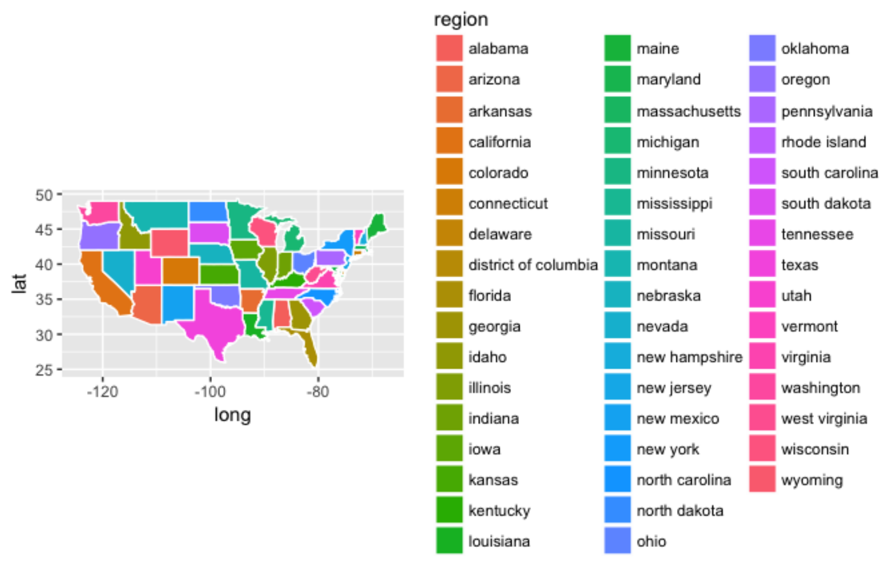
```
# a simple black map of USA with states borders shown
ggplot() + geom_polygon(data = states, aes(x=long, y = lat), group = states$group) +
  coord_fixed(1.5) + ggtitle("Map of the United States with states borders")
```



Now we could see the states borders clearly shown on the map.

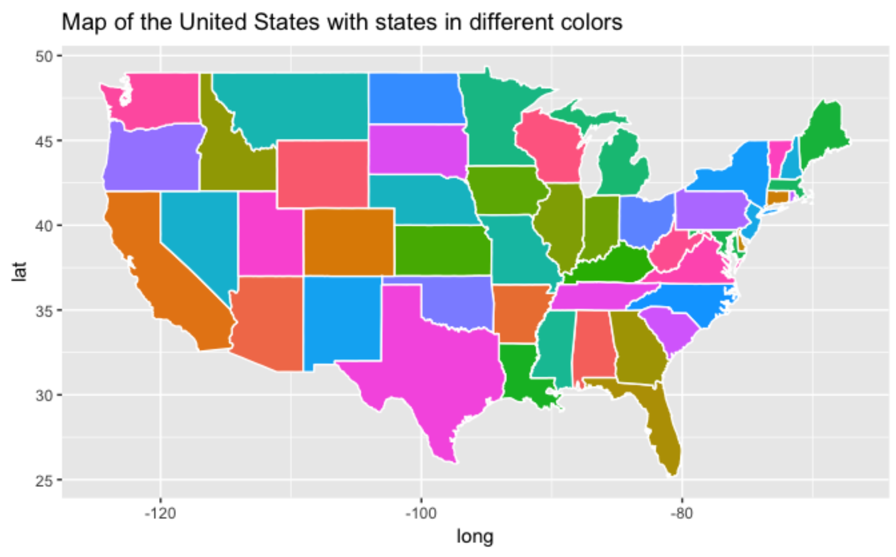
However, only adding the borders between states is not enough. To make our map look better, we could use different colors to represent each state. We could achieve this goal by adding a new parameter "fill" into our `"geom_polygon"` function.

```
# a simple black map of USA with states in different colors
ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white") + coord_fixed(1.3)
```



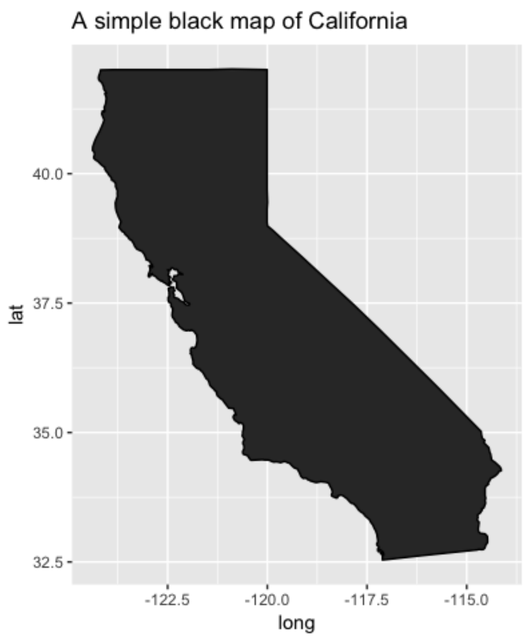
However, the guide on the side takes even more space than the plot itself. To remove the guide, we could add `"guides(fill=FALSE)"` into the ggplot.

```
# a simple black map of USA with states in different colors after removing guides
ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group), color = "white") +
  coord_fixed(1.3) +
  guides(fill=FALSE) + ggtitle("Map of the United States with states in different colors")
```



Let's take a closer look at the map of California.

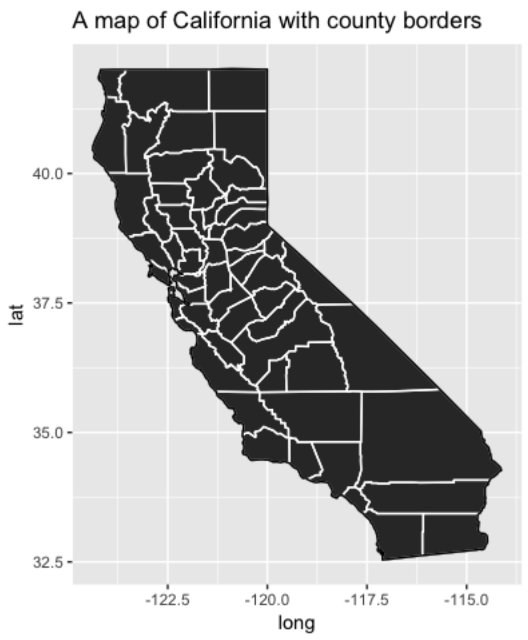
```
#Simple black map of the State of California
ca <- ggplot(data = subset(states, region == "california"), mapping = aes(x = long, y = lat, group = group)
  coord_fixed(1.3) +
  geom_polygon(color = "black") + ggtitle("A simple black map of California")
ca
```



This time, suppose we want to add the border between each counties in california. Recall what we did last time, we utilized the `"state"` dataset in `"maps"`. However, we can't apply this data set on California. We do have another option.

There is a subset called `"county"` in the `"map_data"` package which keeps track of each county of the United States. And we could extract the counties of California by using `"subset"` again.

```r
#A map of California with county borders
counties <- map_data("county")
ca_county <- subset(counties, region == "california")
ca + geom_polygon(data = ca_county, fill = NA, color = "white") +
  geom_polygon(color = "black", fill = NA) + ggtitle("A map of California with county borders")
```



Conclusion

After reading this post, I hope you have some thought about how to use `"maps"`, `"map_data"` and `"ggplot2"` to make maps. Different from the approach we used during lab, this way of plotting maps takes less effort. A lot of useful data you might need is already included in the data packages, and what you need is just subsetting the data set you need instead of serching a data frame online. Besides that, we could add a lot of personalized features on our map, such as color and border. I hope you could use these powerful tools in the future.

References

1. data set in Lab 11
2. map_data package documentation
3. maps package documentation
4. Making maps with R
5. geom_polygon in R
6. USA map using map_data
7. Special maps in R