GGMaps: Theme Options & University Exploration

Introduction:

Have you ever wondered what other data visualizations you can create in R? So far in this course, we have explored various packages such as *readr* to read in files, *dplyr* to manipulate data, and *ggplot* to graphically display data. The exploratory example at the end of the post will make use of all 3 packages, in addition to a new one that we have not learned, *ggmap*. While we have displayed scatterplots, horizontal & vertical barplots, and starplots, I would now like to explore a new type of data visualization: **maps**.

My motivation for dedicating this blog post to maps is because it's a topic we have not discussed in this course yet, and it was a discovery process for me to research the map functions in R and learn about different use-cases for geographical data. And also, maps are just fun!

To begin, I will introduce the *ggmap* package, and display a few examples. **Then**, we will dive into an *exploratory example* to map states by their total number of universities/colleges.

What is the GGMap Package?

The *ggmap* package includes a function called *get_map* to display maps of specified locations from your choice of sources: Google Maps, OpenStreetMap, or Stamen Maps. Arguments for *get_map* include, but are not limited to:

- Location: the longitude & latitude coordinates. You can access these numbers from Google Maps by right-clicking the point of interest and selecting "What's here?"
- Zoom: values range from 3 (continent) to 21 (building), default is 10 (city). Cannot display map of the whole world.
- Maptype: this is a character string specifying the theme such as
    - "satellite" (Google Maps)
    - "terrain" (Google Maps)
    - "hybrid" (Google Maps)
    - "toner" (Stamen Maps)
    - "watercolor" (Stamen Maps)
    - "terrain" (Stamen Maps)
    - Positive integers (Cloudmade Maps)
- Source: "google" (Google Maps), "stamen" (Stamen Maps), "cloudmade" (CloudMade Maps), "osm" (OpenStreetMap)

You can find more details regarding get_map arguments with **?get_map**

Simple Examples:

In order to use any package, we must first install it in R (only need to do this once) and load it with the function *library()*.

```
#install.packages("ggmap")
library(ggmap)
```

```
## Loading required package: ggplot2
```

```
#these packages will be used in the exploratory example
library(ggplot2)
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.4.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

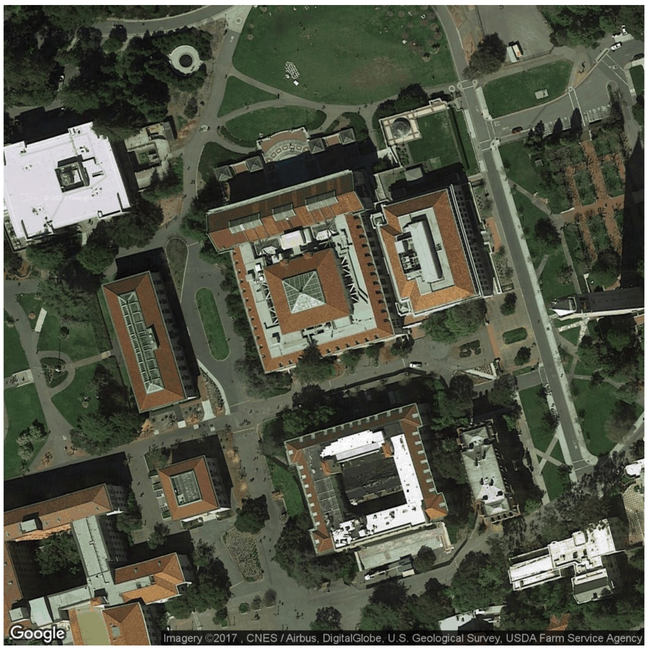
This first example is a **satellite** view of *'Doe Library'* to get a visualization of the real landscape. However there are no labels to know what you are viewing:

```
doe_satellite <- get_map(location = c(lon =  -122.259497, lat = 37.871960),
    scale = "auto",
    color = "color",
    source = "google",
    maptype = "satellite",
    zoom = 18)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=37.87196,-122.259497&zoom=18&size=640x640&s
cale=2&maptype=satellite&language=en-EN&sensor=false
```

```
ggmap(doe_satellite,
    extent = "device",
    ylab = "Latitude",
    xlab = "Longitude")
```

```
## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```


Google
Imagery ©2017 , CNES / Airbus, DigitalGlobe, U.S. Geological Survey, USDA Farm Service Agency

The second example is a **roadmap** of the surrounding area with labels and eliminates the clutter of objects. This is navigation-style type of map that you may be most used to seeing on your mobile device, but the viewer does not have a clear sense of the real landscape:

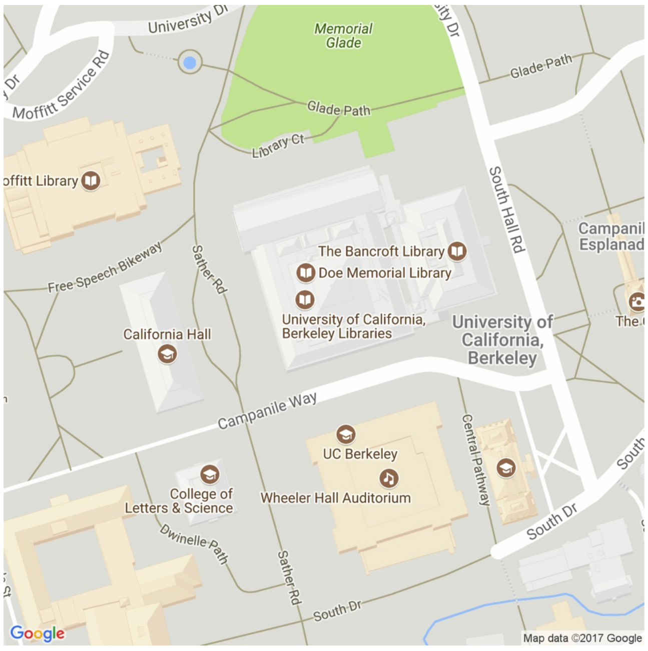
```
doe_terrain <- get_map(location = c(lon =  -122.259497, lat = 37.871960),

    color = "color",
    source = "google",
    maptype = "roadmap",
    zoom = 18)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=37.87196,-122.259497&zoom=18&size=640x640&s
cale=2&maptype=roadmap&language=en-EN&sensor=false
```

```
ggmap(doe_terrain,
    extent = "device",
    ylab = "Latitude",
    xlab = "Longitude")
```

```
## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```



Finally, here is a **hybrid** that is in between the roadmap and satellite map types, which labels the real landscape:

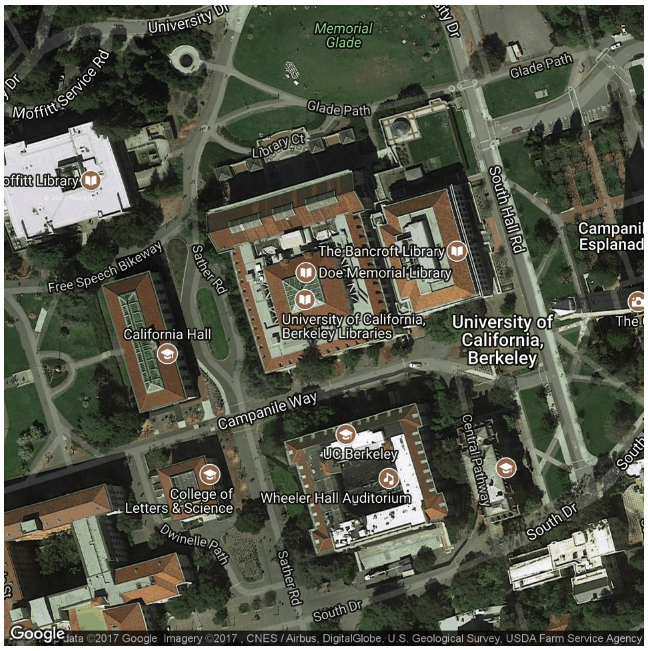
```
doe_terrain <- get_map(location = c(lon =  -122.259497, lat = 37.871960),

    color = "color",
    source = "google",
    maptype = "hybrid",
    zoom = 18)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=37.87196,-122.259497&zoom=18&size=640x640&s
cale=2&maptype=hybrid&language=en-EN&sensor=false
```

```
ggmap(doe_terrain,
    extent = "device",
    ylab = "Latitude",
    xlab = "Longitude")
```

```
## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```



Just for fun, here is the San Francisco Bay Area in a **watercolor** version, which is a rough mapping of land masses, bodies of water, and transportation channels without any labels. Furthermore, including **extent = "panel"** will display the longitude and latitude as axis labels:

```
sf_bay_watercolor <- get_map(location = c(lon =  -122.259497, lat = 37.871960),

    color = "color",
    source = "stamen",
    maptype = "watercolor",
    zoom = "auto")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=37.87196,-122.259497&zoom=10&size=640x640&s
cale=2&maptype=terrain&sensor=false
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/162/394.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/163/394.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/164/394.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/165/394.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/162/395.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/163/395.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/164/395.jpg
```

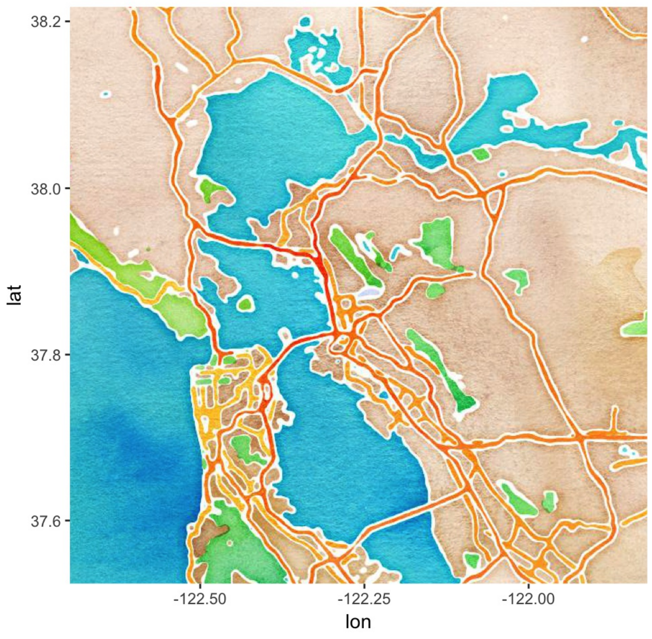
```
## Map from URL : http://tile.stamen.com/watercolor/10/165/395.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/162/396.jpg
## Map from URL : http://tile.stamen.com/watercolor/10/163/396.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/164/396.jpg
```

```
## Map from URL : http://tile.stamen.com/watercolor/10/165/396.jpg
```

```
ggmap(sf_bay_watercolor,
    extent = "panel",
    ylab = "Latitude",
    xlab = "Longitude")
```



Simplified Alternative: 'qmap'

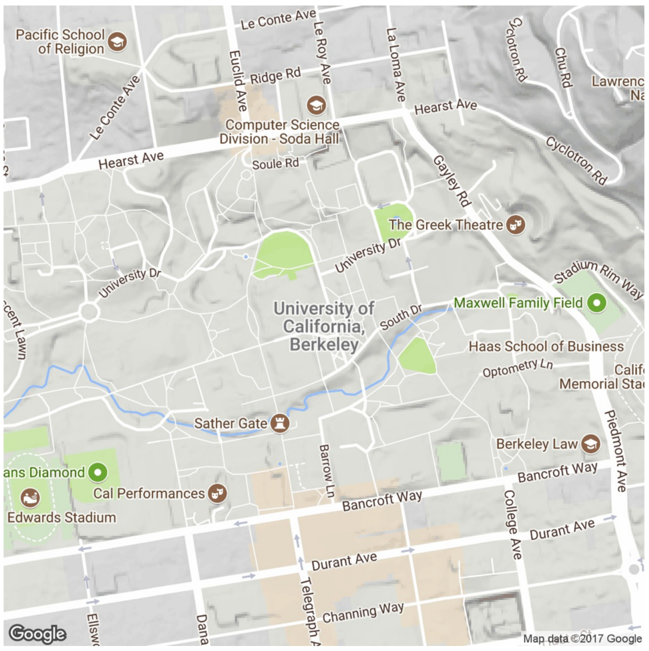
As you may have noticed from the code above, I used specific longitude and latitude coordinates. If you wanted a quick map display without troubling yourself to find a map source and longitude and latitude coordinates, you can use **qmap**.

```
qmap(location = "uc berkeley", zoom = 16)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=uc+berkeley&zoom=16&size=640x640&scale=2&ma
ptype=terrain&language=en-EN&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=uc%20berkeley&sensor=false
```

```
## Warning: `panel.margin` is deprecated. Please use `panel.spacing` property
## instead
```



Here's how it works: qmap is a *wrapper* for get_map and ggmap. The *wrapper* calls another function but provides an alternative, often more convenient, syntax or interface. Usually the wrapper has the same arguments, but different defaults.

- For example, **read.csv** is one wrapper to **read.table**. You only need to choose the right file name and the other arguments in read.table will simplified and taken care of.

As you can see in the code with *qmap* above, there's no need to specify latitude or longitude. *ggmap* uses the string inputs with the "location" parameter when creating a map.

Now we will be moving onto a use case of *ggmap* where we will be exploring and visualizing the states in the country that have a high number of universities/colleges.

Exploratory Example: # of Universities By State

This is an example of a cluster visualization of universities across the country using data from UnivSearch. The end output is a map visual representing the total number of universities/colleges in a state as depicted by the size of the circles.

I have intentionally only included the code blocks without executing them (using eval=FALSE, echo=TRUE) for this exploratory example because I have reached my query limit for Google Maps API, so it will not generate the coordinates for all states when knitting, and therefore will not show the proper data tables. I have included descriptive code comments, and displayed the output as a pdf I sank earlier.

The code below reads in the data file, which includes columns for the State/Possession, Abbreviation, and Total Number of Universities per State. I excluded non-states (total 9) like D.C. and Puerto Rico to visualize just states, not territories or special districts.

```
#read in the data file
by_state <- data.frame(read.csv("by_state.csv"), stringsAsFactors = FALSE)
#Not sure why stringsAsFactors = FALSE didn't keep characters from being converted to factors, so I used an altern
ative.
by_state$Abbreviation <- as.character(by_state$Abbreviation)
by_state$State_Or_Possession <- as.character(by_state$State_Or_Possession)

#Excludes territories and possessions, so we are only looking at universities lcoated within the states
by_state <- by_state[by_state$Abbreviation != "AS" & by_state$Abbreviation != "DC" & by_state$Abbreviation != "FM"
& by_state$Abbreviation != "GU" & by_state$Abbreviation != "MH" & by_state$Abbreviation != "MP" & by_state$Abbrevi
ation != "PR" & by_state$Abbreviation != "PW" & by_state$Abbreviation != "VI", ]
```

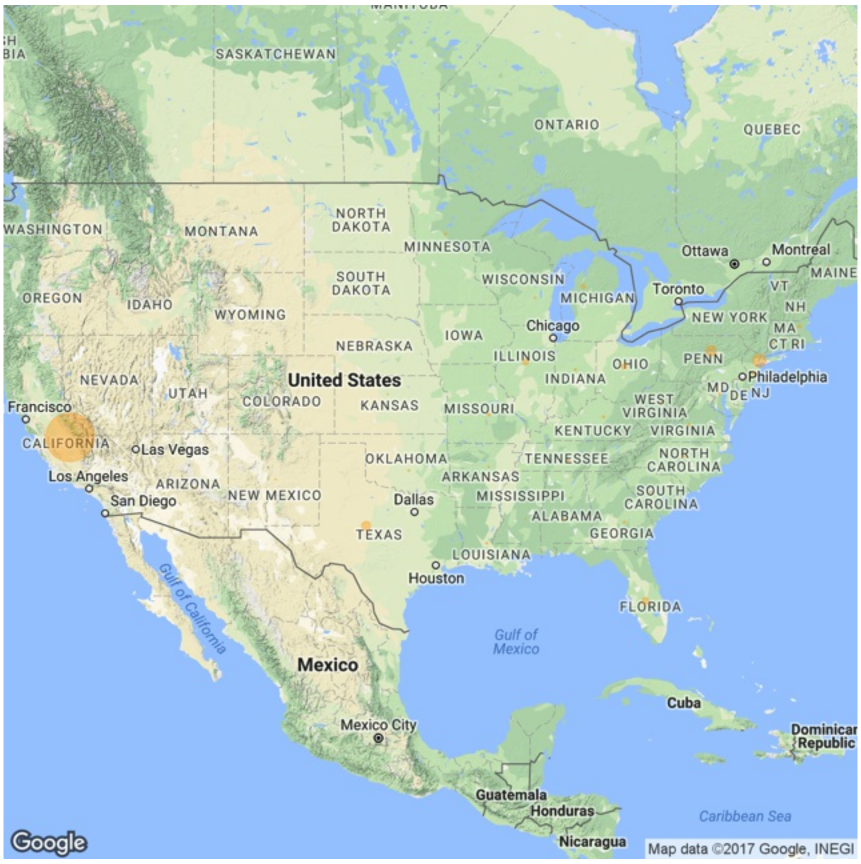
The code below uses ggmap's **geocode** function, which utilizes GoogleMaps API to pull longitude and latitude coordinates of each state.

```
by_state <- mutate(by_state, lon = rep(0, nrow(by_state)) , lat = rep(0, nrow(by_state)))
by_state
#Geocode is a ggmap function that uses Google Maps API to pull coordinates.
#Unfortunately, I have reached my query limit so I cannot generate all the coordinates after re-running.
for (i in 1:nrow(by_state)) {
  latlon = geocode(by_state[i,1])
  by_state$lon[i] = as.numeric(latlon[1])
  by_state$lat[i] = as.numeric(latlon[2])
}
```

The following code overlays a ggplot scatterplot onto the map of the USA that uses aesthetic attributes of longitude and latitude coordinates of each state. Furthermore, I scaled the size of the circles by the proportion of total universities by state, and divided by a factor of 10 to make sure the circles would fit on the graph.

```
usa_center <- as.numeric(geocode("United States"))
#Map Background
usa_map <- ggmap(get_googlemap(center=usa_center, size = c(640,640), scale=2, zoom=4), extent="device")
circle_scale_amt <- prop.table(by_state$Total)

#ggplot overlay
#png(file = 'usa_map.png', pointsize = 20, width = 640, height = 640)
usa_map + geom_point(data = by_state, aes(x=lon, y=lat), col="orange", alpha=0.4, size = (by_state$Total*circle_sc
ale_amt)/10)
#dev.off()
```



Conclusions:

As you can see, the **top 3 states** with the highest number of universities/colleges are **California, New York, Pennsylvania** at 1246, 632, and 544 universities, respectively. Other top states include Texas, Florida, Illinois, Michigan, and Ohio. Though you may not clearly see circles within other states, the circles exist if you look closely. The scaling factor I used allows the viewer to clearly see the states with higher number of universities.

While we could have obtained the same conclusions by analyzing the data table, a map visualization geographically pinpoints the data, so even a visitor from outside the U.S. could clearly understand.

A limitation I noticed from this visualization is that the Google Maps longitude and latitude coordinates may not depict the location the most accurately. For example, the 2nd largest circle located close to Philadelphia may initially be hard to interpret as representing the state of New York.

Key-Message Take Aways:

Now that you've made it through the whole post, I hope you have a better understanding how to use the ggmap package and learned about a use-case through the university totals exploratory example.

Maps are useful in visualizing any type of population data whether for public health, environmental science, city landscaping, and more. By combining a map visualization with quantitative data (whether counts, means, etc), people with and without geographical familarity with the region can better understand the data. Another type of map that I did not cover in this blogpost that you could explore are heat maps and contour maps. Thanks for reading!

References:

1. University by State Data Set: http://www.univsearch.com/state.php
2. GGMap Introduction: https://www.r-bloggers.com/google-maps-and-ggmap/
3. Definition of a Wrapper: https://stat.ethz.ch/pipermail/r-help/2008-March/158393.html
4. Exploratory Visualization: https://blog.dominodatalab.com/geographic-visualization-with-rs-ggmaps/
5. Google Maps: https://rpubs.com/nickbearman/r-google-map-making
6. Google Geocode Limit: https://gis.stackexchange.com/questions/15052/avoiding-google-maps-geocode-limit
7. Heat Maps: https://plot.ly/r/heatmaps/
8. Contour Maps: https://plot.ly/r/contour-plots/