post2: Visualizing data using R——how to draw different maps

Chuchu Gao

Content:

1. Motivation and Introduction:

Earlier this month, we were introduced on how to use R to draw maps using library “RgoogleMaps” and library “ggmap”. It is very interesting to me that we can obtain the map of a specific area from an online resource, google or Data Science Toolkit, and add some value onto it. In the lab, we have a dataset of mobile food locations in San Francisco, and we were able to mark these locations in the map. Inspired by this interesting example, I was wondering how to draw a map that is more general, as well as how to apply what we have learnt to analyze other important datasets and visualize the information. After some research, I want to introduce a new library called “maps” and some further applications of “ggmap”.

```
#The libraries we will be using in this post are "dplyr", "maps" and "ggmap". You should have "dplyr" and "ggmap" installed. If you didnt have the library "maps" installed, you can delete "library(maps)". I will provide the code to install and load "maps" later in this post.
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
```

```
library(ggmap)
```

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

```
library(maps)
```

Note: In Lab11, Professor Sanchez mentioned that it is possible that you run into some issues with “ggmap” (and “ggplot2”). He said that there are a couple of conflicting bugs in some versions of these packages. Here is the code he provided in case you have some cryptic errors, you may switch to an older version of “ggplot2”

```
# skip this part (come back if you run into some error messages)
# (go back to a previous version of ggplot)
devtools::install_github("hadley/ggplot2@v2.2.0")
devtools::install_github("dkahle/ggmap")
```

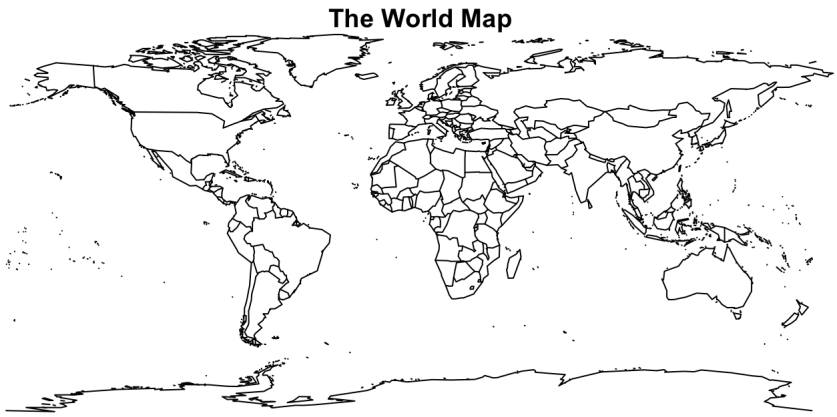
2. Draw maps using library “maps”

“maps” is the library used to draw maps of countries and regions of the world. You can install this package by running the following code.

```
install.packages("maps")
library("maps")
```

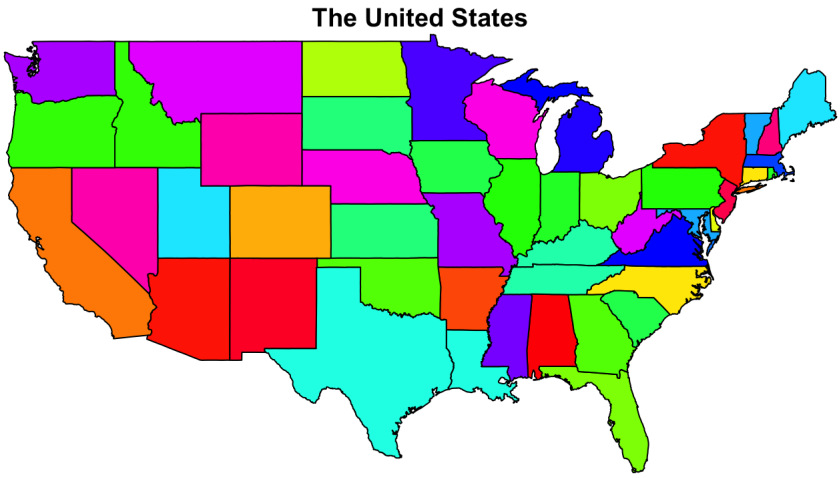
“maps” and “ggmap” generates different types of maps. While “ggmap” applies online sources to provide detailed maps, we can use “maps” to draw the outline of the regions of the world and omit the unneccessary information. Let’s start by drawing a world map.

```
#World map
map(database = "world", mar = c(1,1,1,1))
title("The World Map")
```



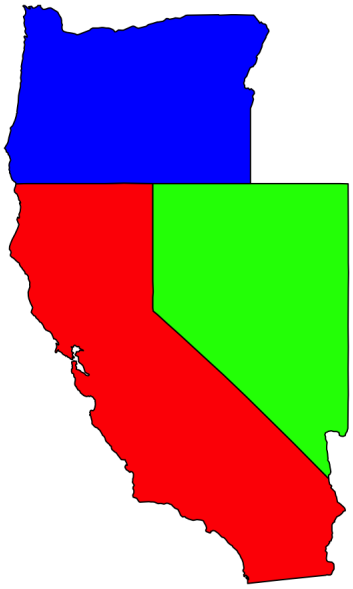
We drew a world map by outling all the countries and regions. “Mar” decides how much margin we want to have on each side of the graph. We can also fill the regions with different colors to make it looks better.

```
#the United States map
map(database = "state",  fill = TRUE, col = rainbow(33),mar = c(1,1,1,1))
title("The United States")
```



Here we are using “state” data to draw the map of United States. Fill is a logical flag says whether to draw lines or fill area with the color set by the “color” vector. We can also draw the map of some specified areas. For example let’s draw a map containing only California, Oregon and Nevada.

```
map("state", regions = c("california","oregon","Nevada"),fill = TRUE, col = rainbow(3),mar = c(1,1,1,1))
```



3. Draw maps using library "ggmap"

Man! These maps look really rough. What if I want a more detailed map? "ggmap" has the access to Google's map data and Data Science Toolkit, an online open-source information server. We can draw a detailed map of a specified region with "ggmap. We can even obtain different types of maps including roadmap, satellite, hybrid, and terrain.

First of all, let me introduce you two very useful functions in "ggmap". This first function is geocode() which can help you to obtain the geographic coordinates of a specific location.

```
#get the geographic information of a specific location
Berkeley_code <- geocode("Berkeley")
```

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

```
Berkeley_code
```

```
##         lon     lat
## 1 -122.2585 37.8719
```

```
#There are more geographic information available for you when you set "output" equals to "more"
Ihouse_code <- geocode("2299 Piedmont Ave, Berkeley, CA", output = "more", source = "google")
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=2299%20Piedmont%20Ave,%20Berkel
ey,%20CA&sensor=false
```

```
Ihouse_code
```

```
##         lon      lat    type loctype
## 1 -122.2514 37.86966 premise rooftop
##                                                  address    north
## 1 international house, 2299 piedmont ave, berkeley, ca 94720, usa 37.87108
##      south      east      west             premise street_number
## 1 37.86838 -122.2501 -122.2528 International House          2299
##             route locality administrative_area_level_2
## 1 Piedmont Avenue Berkeley              Alameda County
##   administrative_area_level_1       country postal_code
## 1                  California United States       94720
```

The second function is mapdist() which can be used to find the distance between two locations. It can also calculate the estimated traveling time based on different transportation methods.

```
#more intesting application of ggmap regarding geographic information
mapdist("Doe Library", "Dwinelle", mode = "walking")
```

```
## by using this function you are agreeing to the terms at :
```

```
## http://code.google.com/apis/maps/documentation/distancematrix/
```

```
## Information from URL : http://maps.googleapis.com/maps/api/distancematrix/json?origins=Doe+Library&destinations
=Dwinelle&mode=walking&sensor=false
```

```
##          from       to   m   km    miles seconds  minutes      hours
## 1 Doe Library Dwinelle 220 0.22 0.136708     160 2.666667 0.04444444
```

Secondly, I want to quickly refresh your memory on how to draw a map with “ggmap”

```
#As an example, I will draw the map for UC Berkeley.
#Step 1. set the range of longitude and latitude of UC Berkeley.
berlon <- c(-122.2400, -122.2590)
berlat <- c(37.8700, 37.8720)
# set "f" equals to the number specifying the fraction by which the range should be extended
ucbbox <- make_bbox(lon = berlon, lat = berlat, f=9)
ucbbox
```

```
##     left   bottom    right      top
## -122.430   37.852 -122.069   37.890
```

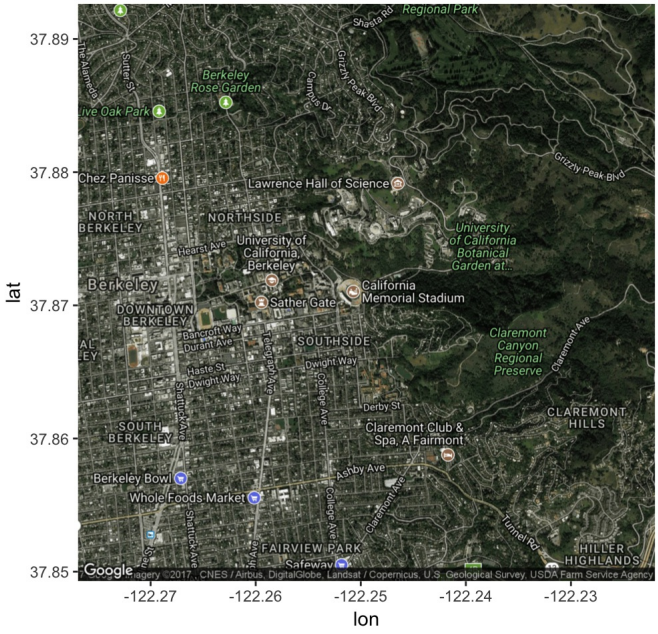
```
#Step 2. Input the range box to get_map() and define the type of map and which source we want to use.
ucb_map1 <- get_map(location = ucbbox, maptype = "hybrid", source = "google")
```

```
## Warning: bounding box given to google - spatial extent only approximate.
```

```
## converting bounding box to center/zoom specification. (experimental)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=37.871,-122.2495&zoom=14&size=640x640&scale
=2&maptype=hybrid&language=en-EN&sensor=false
```

```
#Step 3. Input the map information to ggmap() to get the graph
ggmap(ucb_map1)
```



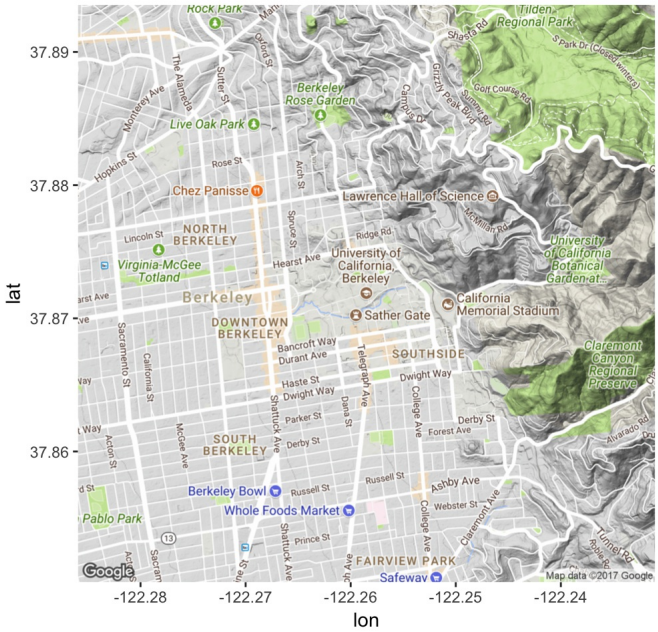
However we don’t have to set the range box everytime. If the geographic information of this location is well defined in Google data, you can simply get the map information by inputting the location’s name.

```
# Get the map by inputting the name of the location
ucb_map2 <- get_map(location = "Berkeley",zoom = 14, maptype = "terrain")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Berkeley&zoom=14&size=640x640&scale=2&mapty
pe=terrain&language=en-EN&sensor=false
```

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

```
ggmap(ucb_map2)
```

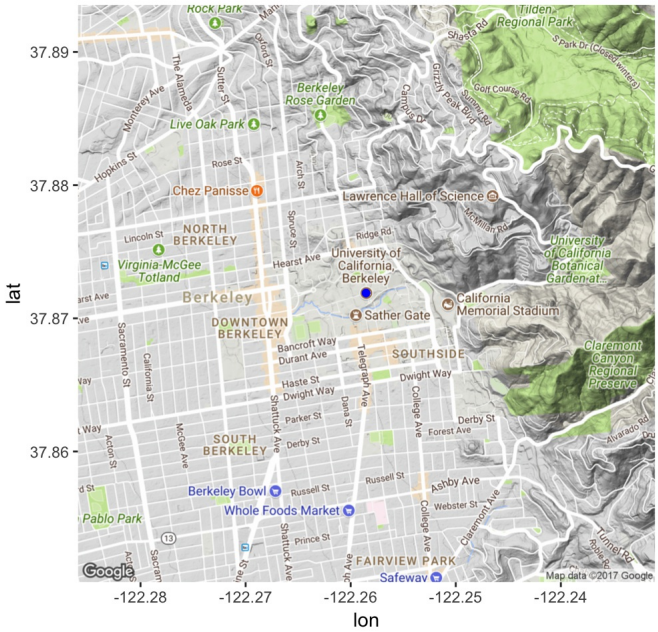


4. Visualizing demographic data

Recall that when we use “ggplot2” to draw a graph, we use the syntax ggplot() to construct a base layer, then we add more layers to construct the graph. geom_histogram(), geom_point(), stat_function and etc are the syntax that we often use. Here in “ggmap”, we can consider that we are replacing the plain base layer in ggplot2 with a map graph, and the rest is the same. That is saying we can also add layers to the map.

The get_map() function returns the map information of an area containing the location that you want to find, However the location is not marked in the map. To mark it in the map, we need to add a layer using geom_point().

```
#We assign the x eauals to the longitude, y eauals to the latitude inside the aes()
#We already got the exact geographic cordinates of Berkeley by using geomcode().
ggmap(ucb_map2)+geom_point(aes(x=Berkeley_code$lon, y=Berkeley_code$lat), color= "blue")
```



What we have talked about seems to be very simple but how can we apply it in the real life? Let’s start with a simple example.

Example 1:

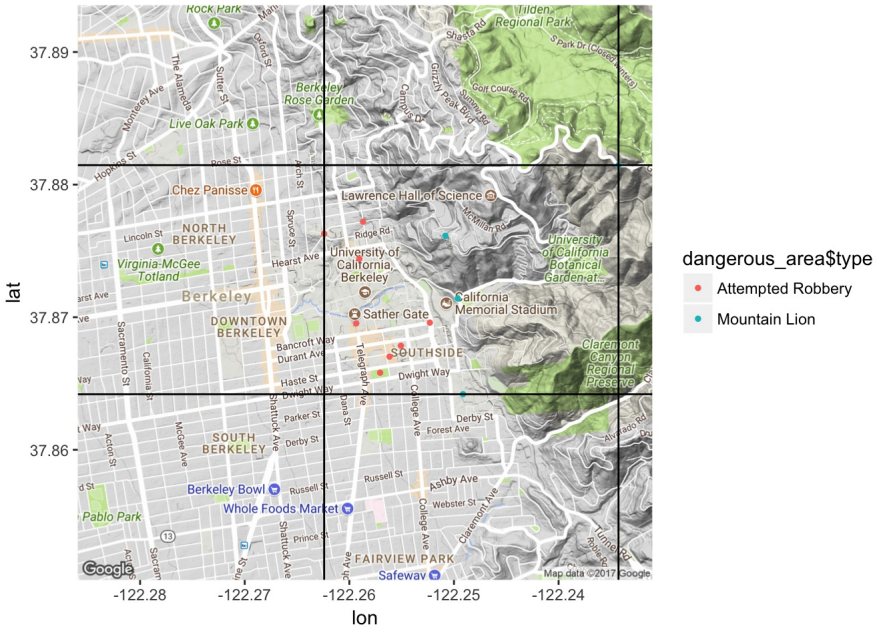
We have all been receiving the alert messages from UCPD regarding the crimes happened around campus. I always want to draw a map with all the crime locations marked. Hopefully by doing so, we can get a better idea of the environment that we are living in.

Step 1. Creating a dataframe containing the location of attempted robberies and mountain lion appeared in the past year. The data information could be found in UCPD’s Alert emails.

```
#Input the longtitude of these locations
dangerous_lon <- c(-122.25082,-122.249635,-122.249138,-122.234273,-122.259336,-122.262406,-122.256160,-122.257058,
-122.255068,-122.252305,-122.259043,-122.258692)
#Input the latitude of these locations
dangerous_lat <- c(37.87614,37.871413,37.864192,37.881470,37.869538, 37.876290 ,37.867029 ,37.865813,  37.867850,
37.869590, 37.874404, 37.877205)
#Categorize them to Attempted Robbery and Mountain Lion
type <- c(rep("Mountain Lion", 4),rep("Attempted Robbery", 8))
type <- as.factor(type)
#Creat the dataset
dangerous_area <- data.frame(type,dangerous_lon, dangerous_lat)
```

Step 2. Mark all the locations on the map. Use different colors for different alert types

```
ggmap(ucb_map2)+geom_point(data=dangerous_area, mapping=aes(x=dangerous_area$dangerous_lon, y=dangerous_area$dange
rous_lat,color=dangerous_area$type), alpha=1,size=1)+geom_hline(yintercept = c(max(dangerous_area$dangerous_lat),m
in(dangerous_area$dangerous_lat)))+geom_vline(xintercept = c(max(dangerous_area$dangerous_lon), min(dangerous_area
$dangerous_lon)))
```




```
# Using geom_vline() and geom_hline() to draw the range of dangerous area
```

From the above graph, we can see that in the past year, the mountain lions are always appears near the east side of the campus which makes sense because mountian lions live in the mountains. Also there are multiple attempted robberies happened inside the campus and on the south side of the campus. Relatively, the northwest side of the campus has less crime reports. In a conclusion, we should try to avoid walking alone during night in the areas that the crimes frequently happened at.

I hope the previous example gives you an idea of how to apply “ggmap” to work with demographic information. I am sure you are now ready to deal with a bigger and more complicated dataset. I am interested in doing a similar analysis of dangerous area in SF. I would like to find in which street robberies are more frequently happen on friday in Mission district for the past few years.

Example 2:

I found an dataset containing the all kinds of incidents happened in city and county of San Francisco in the past few years. You could download the raw data from DataSF website. Click “Export” on the top right corner and choose the file type of be “cvs” to download the data.

After downloading the file “Police_Department_incidents.csv” to your computer. As you might have already noticed, this raw dataset is very large. So let’s first eliminate some unecessary data.

Step 1. cleaning the raw data. **please change the working direction or path accordingly to load the file**

```
#Importing data into R. This data is pretty large. It may take some time to load.
dat <- read.csv("../data/Police_Department_incidents.csv", header = TRUE)
#Observing the Dataset to define what information we want to keep and what information are redundant. Furthermore
we found the type of column "PdDistrict" is character. This Dataset already provides us the geographic coordinates
of the crime locations.
head(dat,3)
```

```
##   IncidntNum     Category                              Descript DayOfWeek
## 1  150060275 NON-CRIMINAL                          LOST PROPERTY    Monday
## 2  150098210      ROBBERY                 ROBBERY, BODILY FORCE    Sunday
## 3  150098210      ASSAULT AGGRAVATED ASSAULT WITH BODILY FORCE    Sunday
##         Date  Time PdDistrict Resolution                    Address
## 1 01/19/2015 14:00    MISSION       NONE       18TH ST / VALENCIA ST
## 2 02/01/2015 15:45 TENDERLOIN       NONE 300 Block of LEAVENWORTH ST
## 3 02/01/2015 15:45 TENDERLOIN       NONE 300 Block of LEAVENWORTH ST
##           X        Y                                Location         PdId
## 1 -122.4216 37.76170  (37.7617007179518, -122.42158168137) 1.500603e+13
## 2 -122.4144 37.78419 (37.7841907151119, -122.414406029855) 1.500982e+13
## 3 -122.4144 37.78419 (37.7841907151119, -122.414406029855) 1.500982e+13
```

```
#Factorize the Police District so that we can extract the Mission district data.
dat$PdDistrict <- factor(dat$PdDistrict, levels=c("BAYVIEW", "CENTRAL", "INGLESIDE", "MISSION", "NORTHERN", "PARK"
, "RICHMOND", "SOUTHERN", "TARAVAL", "TENDERLOIN"))
#Creat a new dataset "Dat" contains only the Mission district data
Dat <- filter(dat,dat$PdDistrict=="MISSION")
```

```
#Subsetting the dataset "Dat" so it only contains the Robberies happened on Friday that have been resolved.
Dat <- filter(Dat, Dat$DayOfWeek == "Friday" & Dat$Category == "ROBBERY" & Dat$Resolution == "ARREST, BOOKED")
Dat <-Dat[, c(2,7,9,10,11,12)]
head(Dat, 3)
```

```
##   Category PdDistrict                     Address         X        Y
## 1  ROBBERY    MISSION SOUTH VAN NESS AV / 17TH ST -122.4173 37.76357
## 2  ROBBERY    MISSION       3100 Block of 16TH ST -122.4224 37.76494
## 3  ROBBERY    MISSION    2800 Block of MISSION ST -122.4183 37.75143
##                              Location
## 1  (37.7635718955893, -122.41733258259)
## 2 (37.7649414890121, -122.422399101985)
## 3 (37.7514284856786, -122.418296094437)
```

Step 2. After got a cleaned dataset, Let draw the map with all the locations marked.

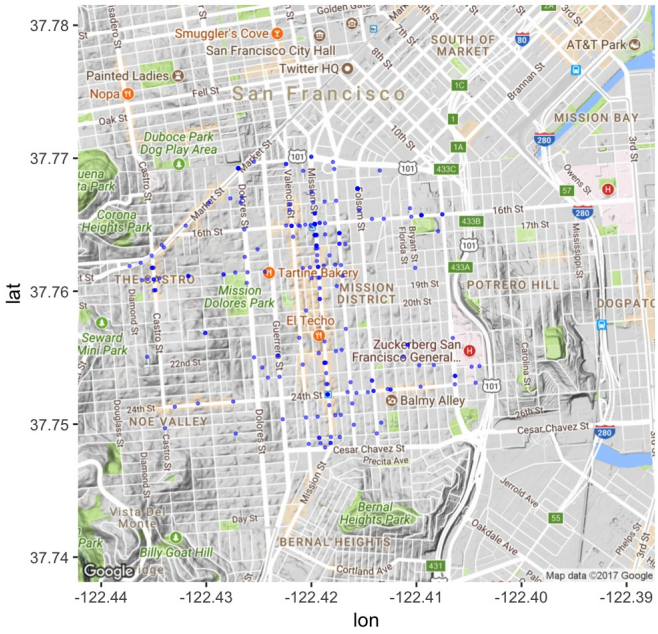
```
#get a map of Mission Strict and add data points of the map. Adjust the "zoom" accordingly to fit the majority of
data points
mission_map <- get_map(location = "Mission, San Francisco",zoom = 14, maptype = "terrain")
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=Mission,+San+Francisco&zoom=14&size=640x640
&scale=2&maptype=terrain&language=en-EN&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=Mission,%20San%20Francisco&sens
or=false
```

```
ggmap(mission_map)+geom_point(data=Dat, mapping=aes(x=Dat$X, y=Dat$Y), color="blue", alpha=0.5,size=0.5)
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```



From the graph, we can say that the Mission St. 16th St. and 24 St. are more dangerous than other areas.

This dataset is very nice because it has already provided us the longitude and the latitude of the crime locations. What if we did not have such important information in the raw dataset? Are we still able to draw the graph? Yes, we can still do it. Do you remember the first function I

introduced earlier in the post? We can get the geographic coordinates of these locations by using geocode(). Let's try it!

Example 3:

Step 1. Let's clean the data and create a new dataset "dat_geocode" just like what we did in the begining in the previous sample.

```
#Select the sub-dataset that we are looking for just like what we did in the begining.
dat_geocode <- filter(dat, dat$PdDistrict == "MISSION" & dat$Resolution == "ARREST, BOOKED" & dat$DayOfWeek == "Fr
iday" & dat$Category == "ROBBERY")
```

```
#observe the data.
head(dat_geocode,5)
```

```
##   IncidntNum Category                           Descript DayOfWeek
## 1  150113244  ROBBERY    ROBBERY ON THE STREET, STRONGARM    Friday
## 2  150181520  ROBBERY ATTEMPTED ROBBERY WITH BODILY FORCE    Friday
## 3  150027463  ROBBERY ATTEMPTED ROBBERY WITH BODILY FORCE    Friday
## 4  140322029  ROBBERY    ROBBERY ON THE STREET, STRONGARM    Friday
## 5  140322029  ROBBERY               ROBBERY, BODILY FORCE    Friday
##         Date  Time PdDistrict     Resolution                    Address
## 1 02/06/2015 02:50    MISSION ARREST, BOOKED SOUTH VAN NESS AV / 17TH ST
## 2 02/27/2015 19:03    MISSION ARREST, BOOKED       3100 Block of 16TH ST
## 3 01/09/2015 19:28    MISSION ARREST, BOOKED    2800 Block of MISSION ST
## 4 04/18/2014 12:15    MISSION ARREST, BOOKED        17TH ST / MISSION ST
## 5 04/18/2014 12:15    MISSION ARREST, BOOKED        17TH ST / MISSION ST
##           X        Y                              Location         PdId
## 1 -122.4173 37.76357  (37.7635718955893, -122.41733258259) 1.501132e+13
## 2 -122.4224 37.76494 (37.7649414890121, -122.422399101985) 1.501815e+13
## 3 -122.4183 37.75143 (37.7514284856786, -122.418296094437) 1.500275e+13
## 4 -122.4195 37.76343 (37.7634292328496, -122.419515708406) 1.403220e+13
## 5 -122.4195 37.76343 (37.7634292328496, -122.419515708406) 1.403220e+13
```

```
# Since this is a very large dataset, for the demostration purpose, I would like plot the first 30 data points onl
y.
dat_geocode <- dat_geocode[1:30,]
```

```
#Omit the longitude and latitude.
dat_geocode <-dat_geocode[, c(2,7,8,9)]
```

```
#Change the Address column to be character variable so geocode() can work with it.
dat_geocode$Address <- as.character(dat_geocode$Address)
geocode <- geocode(dat_geocode$Address)
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=SOUTH%20VAN%20NESS%20AV%20/%201
7TH%20ST&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=3100%20Block%20of%2016TH%20ST&s
ensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=2800%20Block%20of%20MISSION%20S
T&sensor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=17TH%20ST%20/%20MISSION%20ST&se
nsor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "17TH ST /
## MISSION ST"
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=17TH%20ST%20/%20MISSION%20ST&se
nsor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=MISSION%20ST%20/%2019TH%20ST&se
nsor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=CHURCH%20ST%20/%2024TH%20ST&sen
sor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=DOLORES%20ST%20/%2016TH%20ST&se
nsor=false
```

```
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=2800%20Block%20of%2022ND%20ST&s
ensor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "2800
## Block of 22ND ST"
```

```
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=2800%20Block%20of%2022ND%20ST&
sensor=false
```

```
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=100%20Block%20of%20LUCKY%20ST&
sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=3000%20Block%20of%20MISSION%20
ST&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=700%20Block%20of%20SANCHEZ%20ST
&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=700%20Block%20of%20SANCHEZ%20ST
&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=4000%20Block%20of%2024TH%20ST&s
ensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=3200%20Block%20of%2024TH%20ST&
sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=100%20Block%20of%20SANCARLOS%20
ST&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=2000%20Block%20of%20MISSION%20S
T&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=700%20Block%20of%20SOUTH%20VAN%
20NESS%20AV&sensor=false
```

```
## Warning: geocode failed with status OVER_QUERY_LIMIT, location = "700 Block
## of SOUTH VAN NESS AV"
```

```
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=1900%20Block%20of%20MISSION%20
ST&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=1800%20Block%20of%20FOLSOM%20S
T&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=2000%20Block%20of%20MARKET%20S
T&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=1800%20Block%20of%20FOLSOM%20ST
&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=1800%20Block%20of%20FOLSOM%20S
T&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=1800%20Block%20of%20FOLSOM%20ST
&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=18TH%20ST%20/%20CHURCH%20ST&se
nsor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=1100%20Block%20of%20POTRERO%20A
V&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=1800%20Block%20of%20FOLSOM%20S
T&sensor=false
## Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=2100%20Block%20of%20MISSION%20S
T&sensor=false
## .Information from URL : http://maps.googleapis.com/maps/api/geocode/json?address=1800%20Block%20of%20FOLSOM%20S
T&sensor=false
```

```
#get coordinates from google database
dat_geocode <- mutate(dat_geocode, longtitude = geocode$lon,latitude=geocode$lat)
head(dat_geocode)
```

```
##   Category PdDistrict     Resolution                    Address
## 1  ROBBERY    MISSION ARREST, BOOKED SOUTH VAN NESS AV / 17TH ST
## 2  ROBBERY    MISSION ARREST, BOOKED       3100 Block of 16TH ST
## 3  ROBBERY    MISSION ARREST, BOOKED    2800 Block of MISSION ST
## 4  ROBBERY    MISSION ARREST, BOOKED        17TH ST / MISSION ST
## 5  ROBBERY    MISSION ARREST, BOOKED        17TH ST / MISSION ST
## 6  ROBBERY    MISSION ARREST, BOOKED        MISSION ST / 19TH ST
##   longtitude latitude
## 1  -122.4158 37.76430
## 2  -122.4221 37.76507
## 3  -122.4187 37.75199
## 4         NA       NA
## 5  -122.4188 37.76337
## 6  -122.4196 37.76010
```

Step2. Draw the map with all the locations marked.

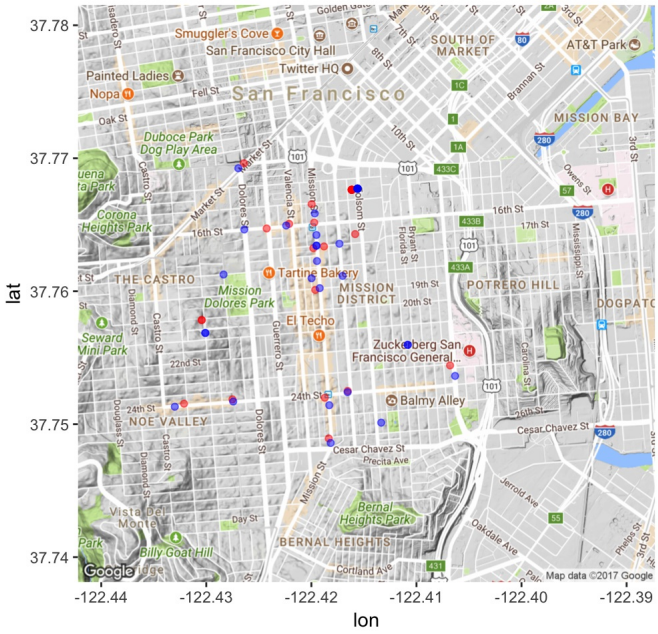
```
#let's draw the map to see if it works
ggmap(mission_map)+geom_point(data=dat_geocode, mapping=aes(x=dat_geocode$longtitude, y=dat_geocode$latitude), col
or="red", alpha=0.5,size=1.5)
```

```
## Warning: Removed 7 rows containing missing values (geom_point).
```

Step3. It seems that it works well. Let's now compare these points with the data from the original data to see if there is any error.

```
#subset the first 30 data point in "Dat"
dat_30 <- Dat[1:30,]
#Using oringinal dataset's longtitude & latitude to add another layer of points
ggmap(mission_map) + geom_point(data=dat_geocode, mapping=aes(x=dat_geocode$longtitude, y=dat_geocode$latitude), c
olor="red", alpha=0.5,size=1.5) + geom_point(data=dat_30, mapping=aes(x=dat_30$X, y=dat_30$Y), color="blue", alpha
=0.5,size=1.5)
```

```
## Warning: Removed 7 rows containing missing values (geom_point).
```



The red points are the locations we got from geocode(). The blue points are the locations we got from the original dataset. From the graph, we can see that for most of the data points, the geographic coordinates we got from geocode() is pretty accurate, however, we do want to keep in mind that there are some minor error existing.

5.Take-home message an conclusion

I hope that now you have a better understanding on how to apply "maps" and "ggmap" to draw different kinds of maps for different purpose. Graph is always one of the most important ways to communicate and spread information. Especially, It is an efficient tool to visualize demographic information. In this post, we tried to present the crime density in different areas, however there are much more topics that we could talk about with library "maps", "ggmap" and others. If you are interested in this topic. I recommend you read through the references for more information. I hope this post it helpful to you.

6. Reference

1. maps - RDocumentation
2. ggmap - RDocumentation
3. Police Department Incidents–DataSF
4. *"Introduction to visualising spatial data in R"* from Robin Lovelace, James Cheshire, Rachel Oldroyd and others (you can download the pdf version online for free)

5. Mapping locations in R with the Data Science Toolkit
6. Making Maps with R
7. Mapping The United States Census With { ggmap }: Machine Learning with R