# GIS 3 Lab 7

Ryan (Yigong) Wang

## Introduction

This lab utilizes the first example in the urban anlytics lab "Data Viz 4 - Mapping Flows." The first exmaple in the lab demonstrates mapping flows of San Francisco data, and upon further exploration of the dataset used in the lab, I find that there are also data for the Silicon Valley (coded as San Jose, Mountainview, and Palo Alto in the dataset.) I will explore using this portion of the data for this lab.

With interaction data, we can explore the relationship between estimation and representation of routes from co-ordinate pairs, mapping GPS trails and how very large origin-destination flows can be summarized. The lab introduces interaction data for this example in the following way:

> *Many interaction data within cities are simply pairs of origin and destination locations, with some flow recorded between them. A good source of such data which are available within many municipalities relate to the location and flow between bike share docking stations. Many of the operators of these systems now make these data openly available.*

## Silicon Valley Bikeshare Data

We will now read in the September 2015 - August 2016 data from the Bay Area Bike Share, SF, USA. We will just read in the Silicon Valley portion of data.

### Selecting the Data

```
#Read in data
stations <- read.csv("./data/201608_station_data.csv")
trips <- read.csv("./data/201608_trip_data.csv")
```

Each of the stations has various attributes and cover a series of locations within the bay area - in this case, we will subset to only those within San Francisco.

```
head(stations)
```

```
##   station_id                             name     lat     long
## 1          2 San Jose Diridon Caltrain Station 37.3297 -121.902
## 2          3             San Jose Civic Center 37.3307 -121.889
## 3          4            Santa Clara at Almaden 37.3340 -121.895
## 4          5                 Adobe on Almaden 37.3314 -121.893
## 5          6                 San Pedro Square 37.3367 -121.894
## 6          7              Paseo de San Antonio 37.3338 -121.887
##   dockcount landmark installation
## 1        27 San Jose     8/6/2013
## 2        15 San Jose     8/5/2013
## 3        11 San Jose     8/6/2013
## 4        19 San Jose     8/5/2013
## 5        15 San Jose     8/7/2013
```

```
## 6         15 San Jose     8/7/2013
```

Since we are exploring the Silicon valley data, we select the relevant portion:

```
# Limit to Silicon Valley stations
stations <- stations[stations$landmark != "San Francisco",]
```

```
library(ggmap)
```

Now we plot a map to show the locations of all the bike share stations in the Silicon Valley:

```
#Get background map for Silicon Valley
register_google(key = "AIzaSyDJGbhjT7t8E4pb0Zu49Ludj4hg4wt1_mQ")
valley <- get_map(location = c(-122.054, 37.371), zoom = 11,color = 'bw')
```

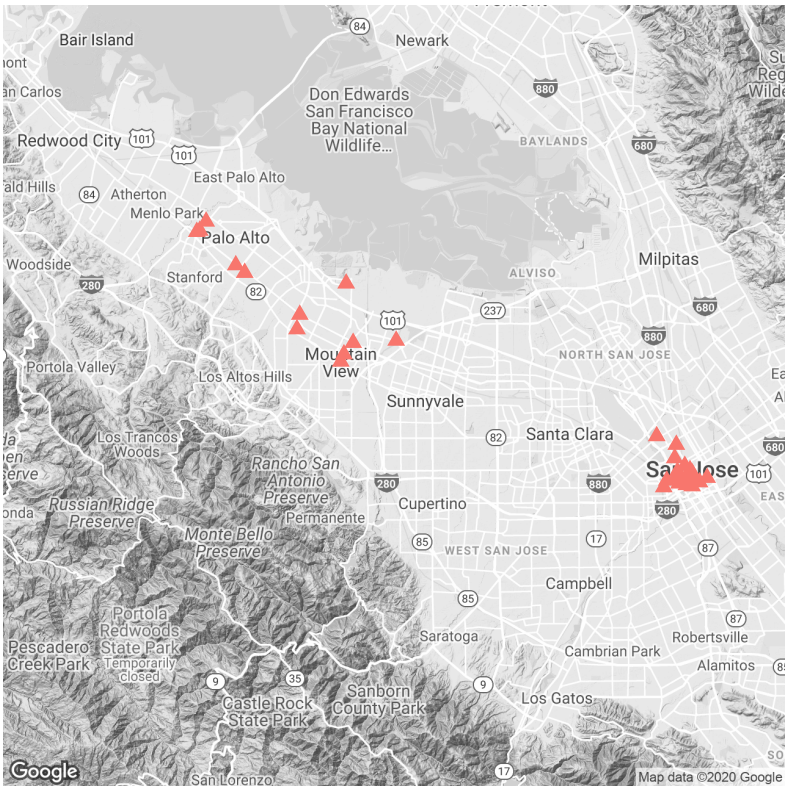
```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=37.371,-122.054&zoom=11&size=640x640&scal
```

```
SJ <- get_map(location = c(-121.889, 37.335), zoom = 14,color = 'color')
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=37.335,-121.889&zoom=14&size=640x640&scal
```

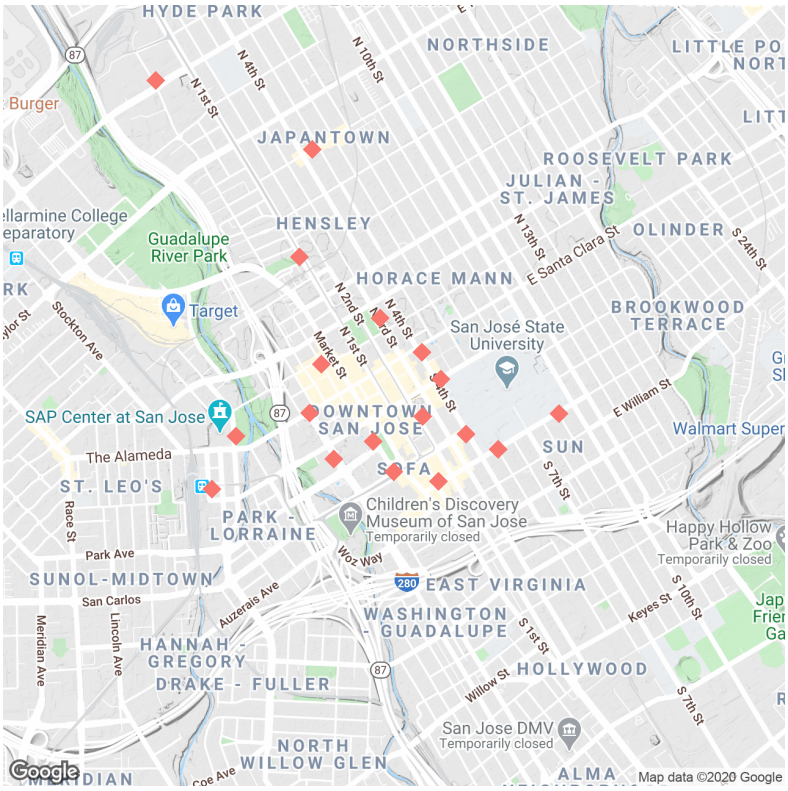
```
#Basic point plot
ggmap(valley) + geom_point(shape=17, size = 2, data = stations, aes(x = long, y = lat, colour = "red")) +
theme_bw() + ggtitle("Bike States in Silicon Valley") +
theme(axis.line = element_blank(),
      axis.text = element_blank(),
      axis.title=element_blank(),
      axis.ticks = element_blank(),
      legend.key = element_blank(),
      legend.position="none",
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.border = element_blank(),
      panel.background = element_blank())
```

## Bike States in Silicon Valley




```
ggmap(SJ) + geom_point(shape=18, size = 3, data = stations, aes(x = long, y = lat, colour = "red")) +
theme_bw() + ggtitle("Bike Stations in San Jose") +
theme(axis.line = element_blank(),
      axis.text = element_blank(),
      axis.title=element_blank(),
      axis.ticks = element_blank(),
      legend.key = element_blank(),
      legend.position="none",
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      panel.border = element_blank(),
      panel.background = element_blank())
```

## Bike Stations in San Jose




Turning now to the trips taken between these origin and destinations, these are ordered within the data frame as a trip per row, with each trip giving a various details including the start and end terminals. Using these data we will create a table of aggregate origin destination flows - however, we will only consider those flows between the stations identified as “San Francisco” and therefore shown on the map above.

```
# View top six rows of the trips data
head(trips)
```

```
##   Trip.ID Duration    Start.Date
## 1  913465      746 9/1/2015 0:10
## 2  913466      969 9/1/2015 0:15
## 3  913467      233 9/1/2015 0:15
## 4  913468      213 9/1/2015 1:29
## 5  913469      574 9/1/2015 1:33
## 6  913470      623 9/1/2015 1:36
##                          Start.Station Start.Terminal
## 1 San Francisco Caltrain 2 (330 Townsend)             69
## 2                        Clay at Battery             41
## 3                      Davis at Jackson             42
## 4                        Clay at Battery             41
## 5                      Steuart at Market             74
## 6       San Jose Diridon Caltrain Station              2
##        End.Date                            End.Station End.Terminal
## 1 9/1/2015 0:23                San Francisco City Hall           58
## 2 9/1/2015 0:31                   Washington at Kearny           46
## 3 9/1/2015 0:19                Commercial at Montgomery           45
## 4 9/1/2015 1:32                        Steuart at Market           74
## 5 9/1/2015 1:42 San Francisco Caltrain 2 (330 Townsend)           69
## 6 9/1/2015 1:47                                Japantown           9
##   Bike.. Subscriber.Type Zip.Code
```

```
## 1    238       Subscriber    94107
## 2     16       Subscriber    94133
## 3    534       Subscriber    94111
## 4    312       Subscriber    94107
## 5    279       Subscriber    94107
## 6    261       Subscriber    95112
```

```
# Get a list of the station IDs within Silicon Valley
s_SV <- unique(stations$station_id)

# Limit trips to those with both origin and destination within the Silicon Valley subset
trips_SV <- trips[(trips$Start.Terminal %in% s_SV) | (trips$End.Terminal %in% s_SV),]

# Create an table with origins and destination pairs
OD_trips_SV <- table(trips_SV$Start.Terminal,trips_SV$End.Terminal)
#View the top six rows
head(OD_trips_SV)
```

```
##
##       2   3   4   5   6   7   8   9  10  11  12  13  14  16  22  23
##   2  67 193 971 354 402 390 118 350  96 289  93 209  26 135   0   0
##   3 148 118  23  31  39  24  13  23  11  18  15   8  20  12   0   0
##   4 932  35  96  14  14  29  11  10  11  11  49   8 206   4   0   0
##   5 372  29   9  58  51  26  53  15  37  22  15  14  17   3   0   0
##   6 396  51  25  37  81  38  31 137  30  16  18 131  53  47   0   0
##   7 322  27  18  17  54  55  60  51  15  24  13  30  17  83   0   0
##
##      24  25  27  28  29  30  31  32  33  34  35  36  37  38  57  60
##   2   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   3   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   4   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   5   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   6   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##   7   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
##
##      70  80  84  88  89
##   2   0  19 314   1   8
##   3   0  12  19   1   8
##   4   0   3  13   0   1
##   5   0   3  65   0   0
##   6   0  50  57   0  17
##   7   1   2  43   0   1
```

```
# Create a data frame of the OD pairs
OD_trips_SV_Narrow <- data.frame(OD_trips_SV)
# Create sensible column names
colnames(OD_trips_SV_Narrow) <- c("Origin","Destination","Trips")
```

We will now identify the top ten most frequently ridden origin-destination pairs

```
# Sorts the trips in decending order
OD_trips_SV_Narrow <- OD_trips_SV_Narrow[order(OD_trips_SV_Narrow$Trips,decreasing = TRUE),]
# Get the top 10 trips
top10 <- OD_trips_SV_Narrow[1:10,]
top10
```

```
##      Origin Destination Trips
## 741      28          27  1186
## 780      27          28  1119
```

```
## 81        2           4   971
## 3         4           2   932
## 785      32          28   854
## 941      28          32   776
## 1030     37          34   546
## 1147     34          37   483
## 161       2           6   402
## 5         6           2   396
```

We will now add origin and destination latitude and longitude co-ordinates by merging with the stations data. First the origin locations:

```
# Add origin co-ordinates
top10 <- merge(top10,stations, by.x="Origin",by.y="station_id", all.x=TRUE)
# Remove unwanted columns
top10 <- subset(top10, select=c("Origin","Destination","Trips","lat","long"))
# Change column names
colnames(top10) <- c("Origin","Destination","Trips","O_lat","O_long")
```

And then the destinations:

```
# Add destination co-ordinates
top10 <- merge(top10,stations, by.x="Destination",by.y="station_id", all.x=TRUE)
# Remove unwanted columns
top10 <- subset(top10, select=c("Origin","Destination","Trips","O_lat","O_long","lat","long"))
# Change column names
colnames(top10) <- c("Origin","Destination","Trips","O_lat","O_long","D_lat","D_long")
```

One of the simplest ways of calculating a route is to use the Google maps API which is implemented in the googleway package.

```
# Install package
#install.packages("googleway")
```

```
# Load package
library(googleway)
```

For this you will need to get a Google maps API key:

```
# Set your key
key <- "AIzaSyDJGbhjT7t8E4pb0Zu49Ludj4hg4wt1_mQ"
```

We will then extract an origin destination pair from our top10 object, and then use the google_directions() function to generate a route - this is then converted to a set of lat lon waypoints using decode_pl():

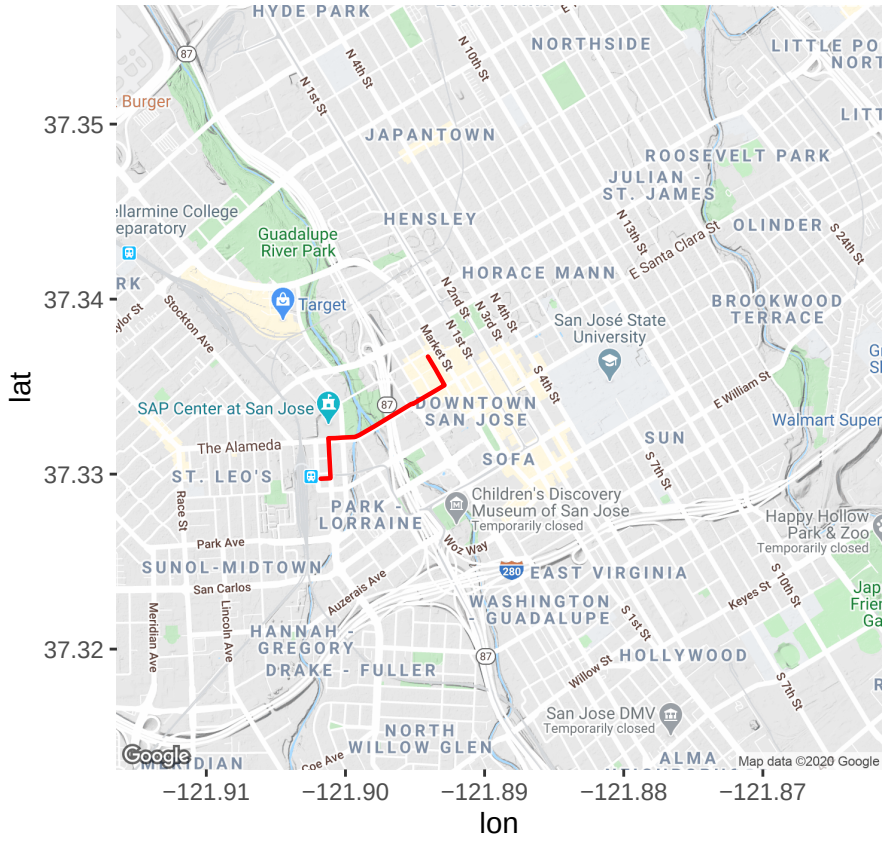
```
# Using the first origin/destination
x <- 2 # You can change this between 1 - 10 to view each of the routes
origin <- c(top10[x,"O_lat"],top10[x,"O_long"])
destination <- c(top10[x,"D_lat"],top10[x,"D_long"])

# get the directions from Google Maps API
res <- google_directions(origin = origin,destination = destination,key = key, mode= "bicycling")

# Convert the results to co-ordinates
df_polyline <- decode_pl(res$routes$overview_polyline$points)

# See the top six rows
head(df_polyline)
```

```
##       lat      lon
```

```
## 1 37.3367 -121.894
## 2 37.3360 -121.894
## 3 37.3351 -121.893
## 4 37.3348 -121.893
## 5 37.3344 -121.894
## 6 37.3342 -121.895
```

These can then be mapped with ggmap:

```
ggmap(SJ) +
  geom_path(aes(x = lon, y = lat), color = "red", size = 0.8, data = df_polyline, lineend = "round")
```




We can extend the above to run a conditional statement with the for() function which does something (in this case, what is in brackets) until a condition is satisfied. Here loop changes the value of x from 1 to the number of rows in the top10 object (i.e. 10), and for each change in x the code between the { and } is run. For loops are very helpful to run a block of code multiple times.

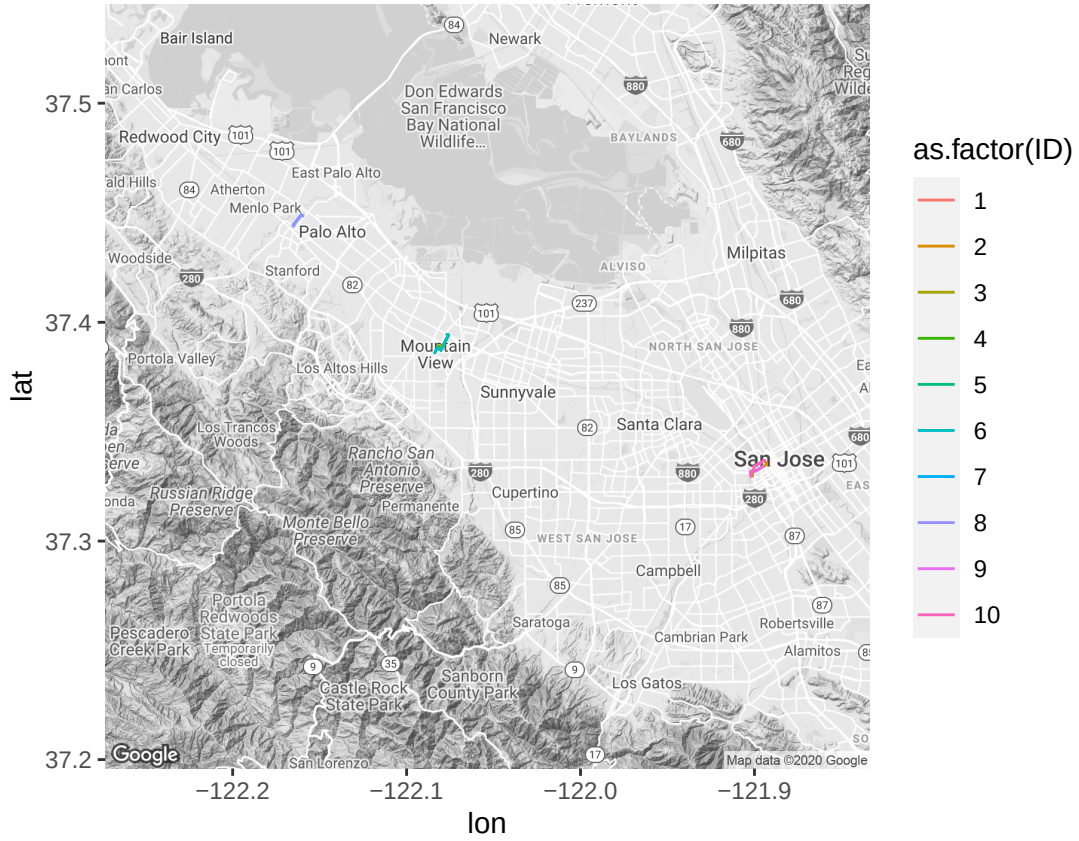
Because x is changed from 1-10 on each run, we can use this value in various helpful ways, firstly to select a particular row from the data frame top10, and second to act as an ID for each set of routes extracted.

```
tmp <-  data.frame(lat = numeric(0), lon = numeric(0), ID = numeric(0), Trips = numeric(0))

for (x in 1:nrow(top10)) {

# Get origins and destinations
origin <- c(top10[x,"O_lat"],top10[x,"O_long"])
destination <- c(top10[x,"D_lat"],top10[x,"D_long"])

# get the directions from Google Maps API
res <- google_directions(origin = origin,destination = destination,key = key, mode= "bicycling")
```

```
# Convert the results to co-ordinates
df_polyline <- decode_pl(res$routes$overview_polyline$points)

# Add a route ID and Trips to the data frame
df_polyline$ID <- x
df_polyline$Trips <- top10[x,"Trips"]

# Append the results to the tmp object
tmp <- rbind(tmp,df_polyline)

}
```

We can now visualize this using the ID as a factor which shows each route as a separate color.

```
ggmap(valley) +
  geom_path(aes(x = lon, y = lat,color = as.factor(ID)), size = 0.5, data = tmp, lineend = "round")
```




To enable some more experimentation with the flow data visualization without having to generate all the potential routes yourself, we have run these already for all origin destination station pairs where the flow was greater than 0. We will load these now:

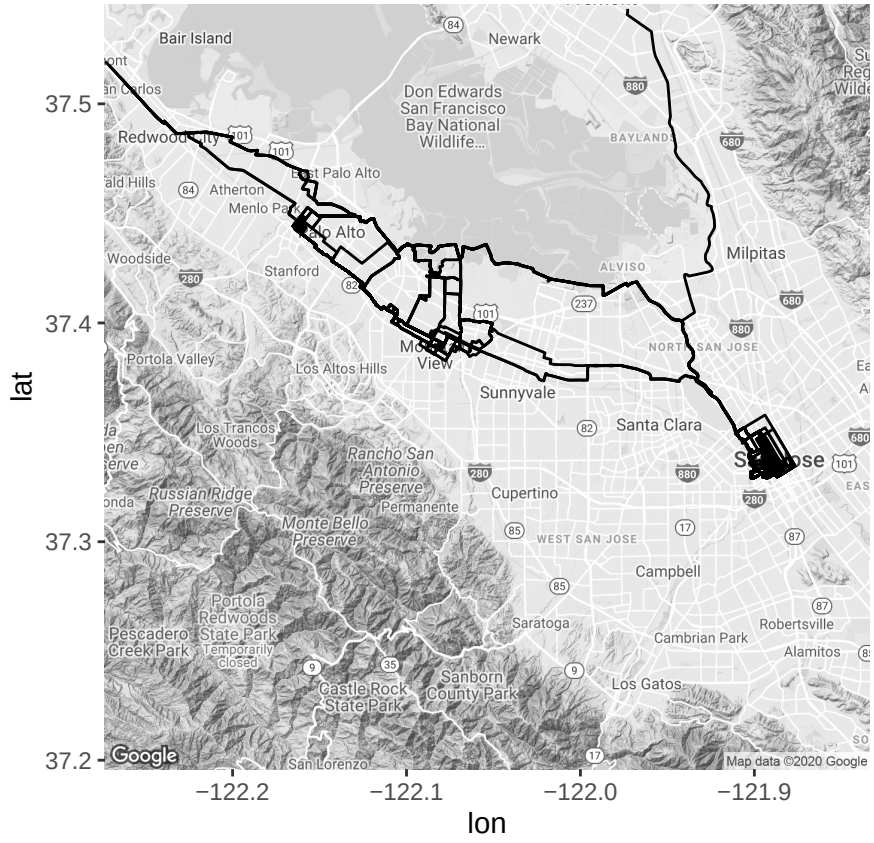
```
# Load flows
load("./data/All_Flows.Rdata")
# Show the top six rows of the table
head(All_Flows)
```

```
##       lat     lon ID Trips
## 1 37.3325 -121.89  1     4
## 2 37.3321 -121.89  1     4
## 3 37.3323 -121.89  1     4
## 4 37.3323 -121.89  1     4
```

```
## 5 37.3323 -121.89  1     4
## 6 37.3328 -121.89  1     4
```

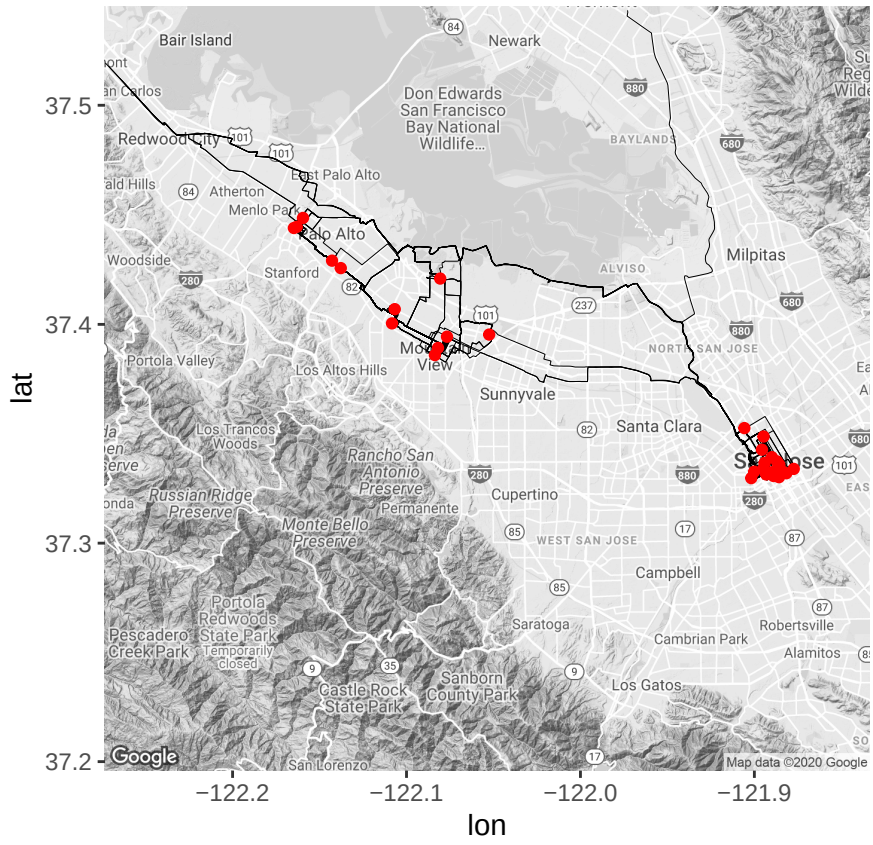
We can now show these on a map - we use the group option within the aes to tell ggmap that these are id that separate the routes, otherwise the whole set of co-ordinates are interpreted as a single route. You can remove these and generate the plot again to see what happens.

```
ggmap(valley) +
  geom_path(aes(x = lon, y = lat, group = ID), data = All_Flows)
```



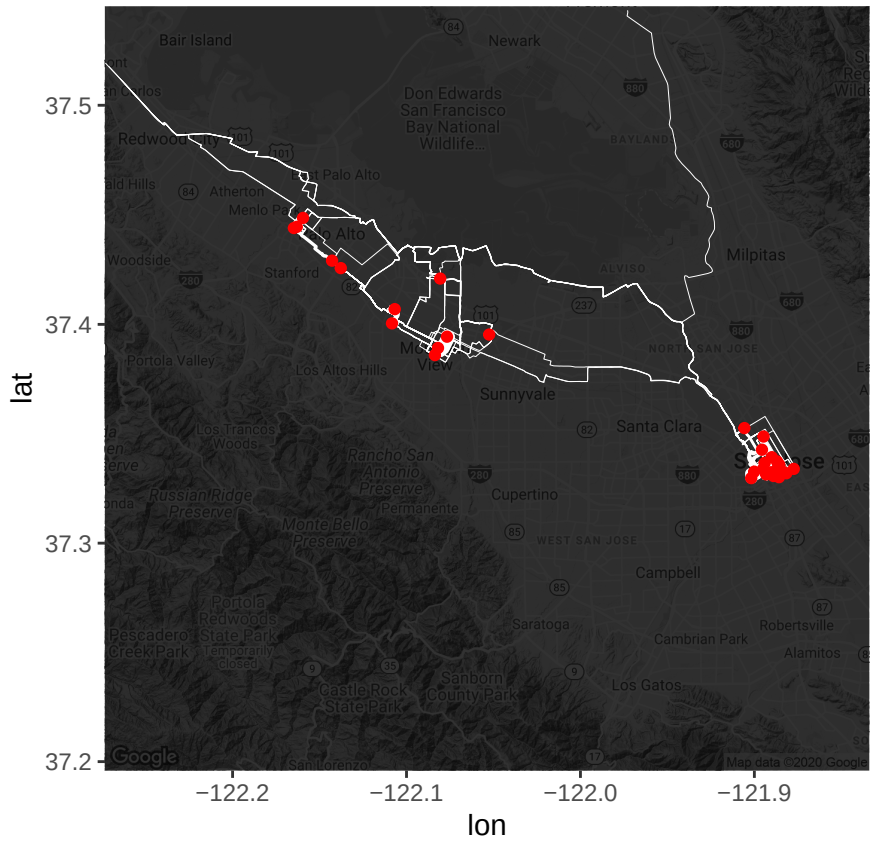

We can now use the trip information to adjust the plot - for example, to scale the routes by the flow volume. We add the size option, but also divide the flows by 1000 to make the line widths an acceptable size. Thicker lines represent greater flows. We have also added the location of the stations in red.

```
ggmap(valley) +
  geom_path(aes(x = lon, y = lat, group = ID), data = All_Flows,size = All_Flows$Trips/1000) +
  geom_point(data=stations, aes(long, lat),colour="red")
```



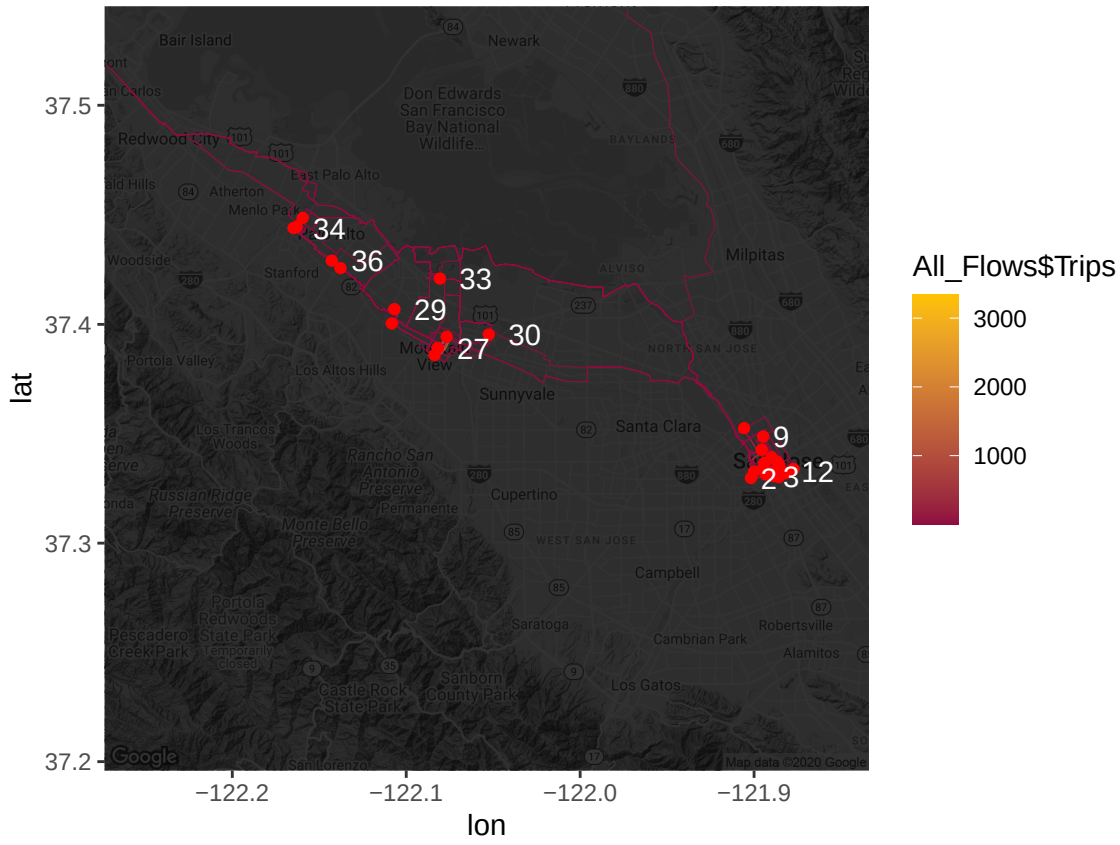

There are a lot of other adjustment options, for example, we can darken the map and change the line color:

```
ggmap(valley,darken = 0.8) +
  geom_path(aes(x = lon, y = lat, group = ID), data = All_Flows, size = All_Flows$Trips/500, colour = "whit
  geom_point(data=stations, aes(long, lat),colour="red")
```



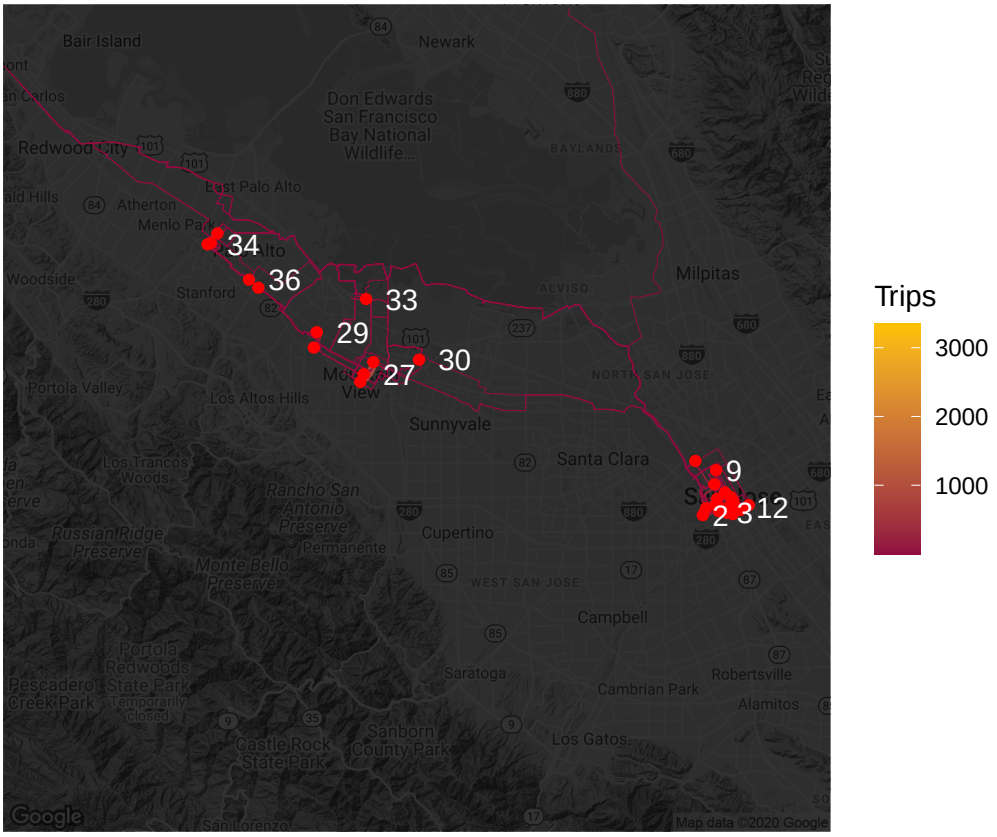

Or color the lines by intensity of flow; plus, we have also added some labels for the station ID using geom_text():

```
ggmap(valley,darken = 0.8) +
  geom_path(aes(x = lon, y = lat, group = ID,colour = All_Flows$Trips), data = All_Flows, size = All_Flows$
  scale_colour_gradient(low="#900C3F", high="#FFC300") +
  geom_point(data=stations, aes(long, lat),colour="red") +
  geom_text(data = stations,aes(x = long, y = lat, label = station_id), check_overlap = TRUE, colour="#FFFF
```




We will now clean up this plot by removing the unwanted elements and changing the title of the legend:

```
ggmap(valley,darken = 0.8) +
  geom_path(aes(x = lon, y = lat, group = ID,colour = All_Flows$Trips), data = All_Flows, size = All_Flows$
  scale_colour_gradient(low="#900C3F", high="#FFC300",name="Trips") +
  geom_point(data=stations, aes(long, lat),colour="red") +
  geom_text(data = stations,aes(x = long, y = lat, label = station_id), check_overlap = TRUE, colour="#FFFF
  theme (
        axis.text = element_blank (),
        axis.title = element_blank (),
        axis.ticks = element_blank ()
        )
```


Bair Island
Newark
Don Edwards San Francisco Bay National Wildlife...
Redwood City
East Palo Alto
Atherton
Menlo Park
Woodside
Stanford
Milpitas
ALVISO
BAYLANDS
NORTH SAN JOSE
Portola Valley
Los Altos Hills
Sunnyvale
Santa Clara
Los Trancos Woods
Rancho San Antonio Preserve
Russian Ridge Preserve
Monte Bello Preserve
Cupertino
WEST SAN JOSE
Campbell
Portola Redwoods State Park
Pescadero Creek Park
Temporarily closed
Saratoga
Cambrian Park
Robertsville
Alamitos
Castle Rock State Park
Sanborn County Park
Los Gatos
San Lorenzo
Google
Map data ©2020 Google
34
36
33
29
30
27
9
2
3
12
Trips
3000
2000
1000