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.

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)
```

Finding the Correct Map Area

Now we plot maps 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')
```

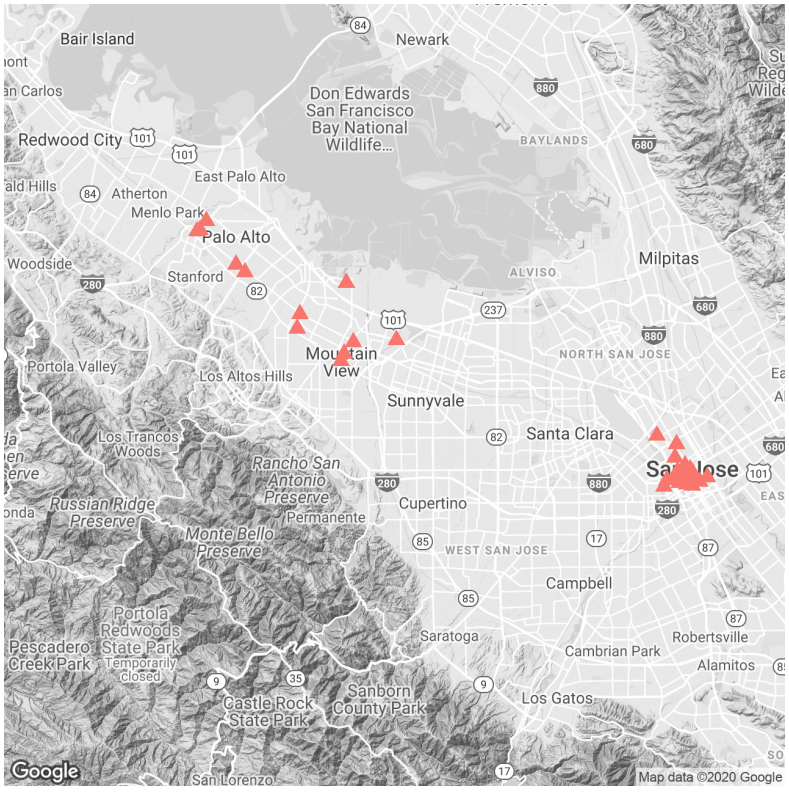
```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=37.335,-121.889&zoom=14&size=640x640&scal
```

```
tech <- get_map(location = c(-122.107, 37.410), zoom = 12,color = 'color')
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=37.41,-122.107&zoom=12&size=640x640&scale
```

```
#Basic point plot
ggmap(valley) + geom_point(shape=17, size = 2, data = stations, aes(x = long, y = lat, colour = "red")) +
theme_bw() + ggtitle("Bike Stations 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())
```

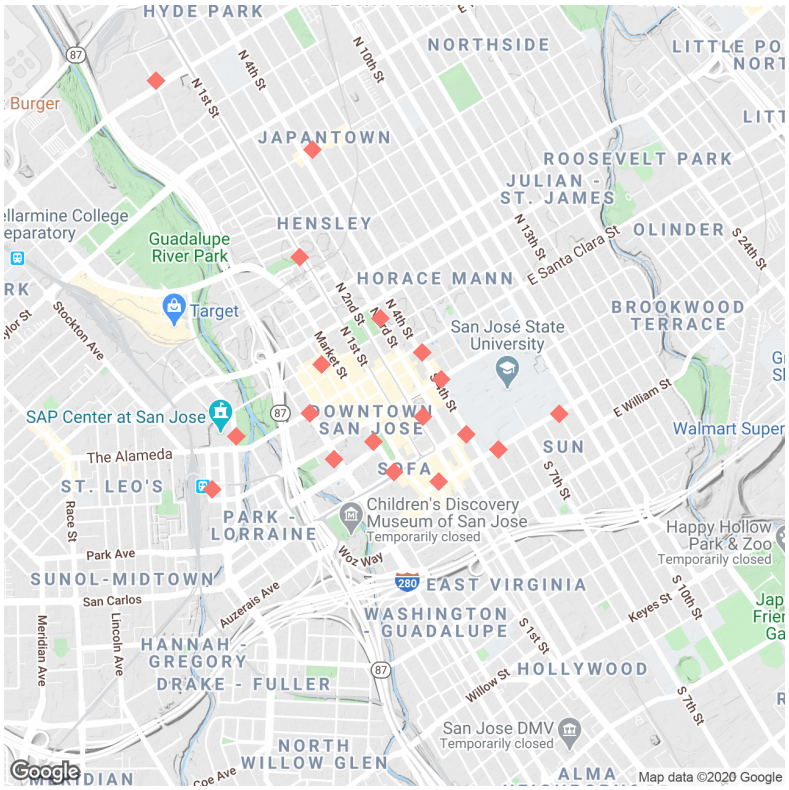
Bike Stations in Silicon Valley




```
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(),
```

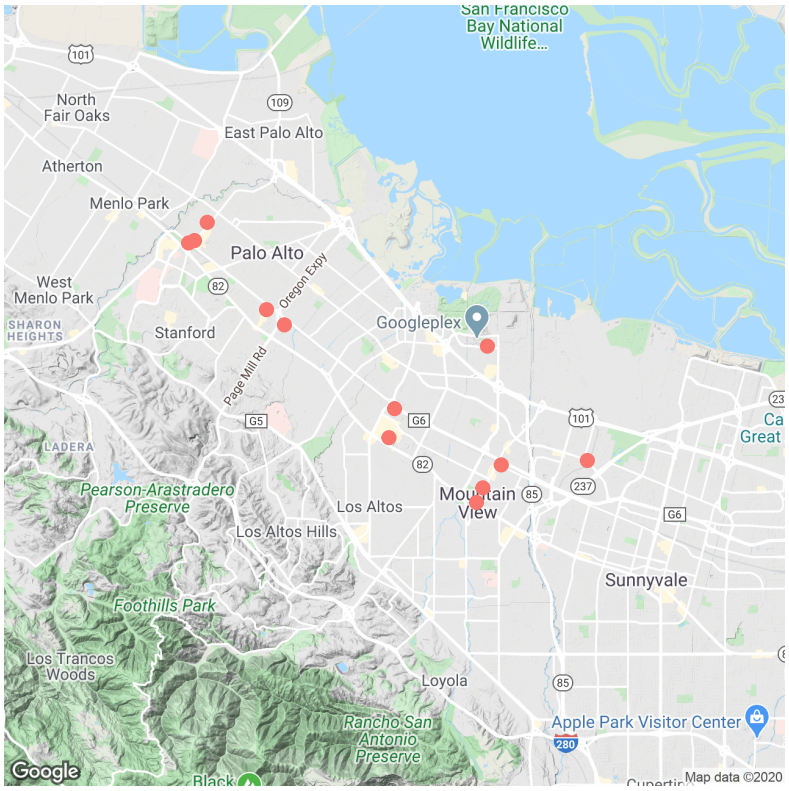
```
      panel.grid.minor = element_blank(),
      panel.border = element_blank(),
      panel.background = element_blank())
```

Bike Stations in San Jose




```
ggmap(tech) + geom_point(shape=19, size = 2, data = stations, aes(x = long, y = lat, colour = "red")) +
theme_bw() + ggtitle("Bike Stations in Palo Alto + Mountain View") +
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 Palo Alto + Mountain View



We need three maps because the Silicon Valley stations are clustered into three cities, with the most prominent being San Jose. The second and thrid map shows the stations in individual cities (Palo Alto and Mountain View are shown in the same map, and this suits the traffic flow analysis done below.)

Exploring Silicon Valley Bikeshare Stations

We then create a dataframe with all stations in the Silicon Valley

```
# 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
```

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.

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

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

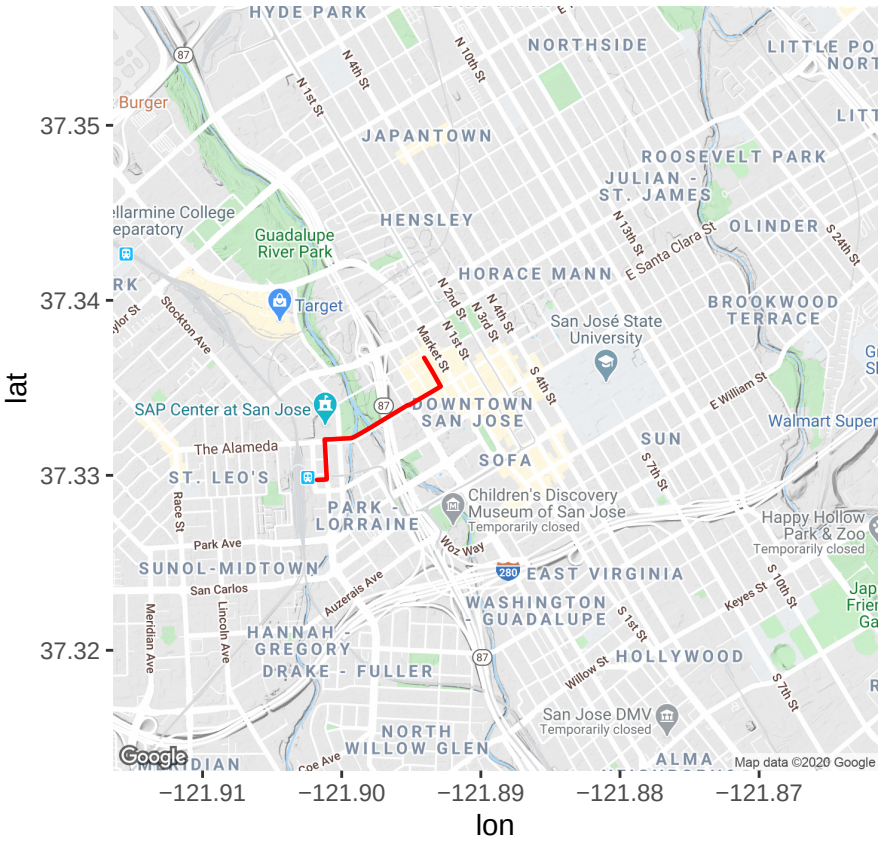
```
# Set your key
key <- "AIzaSyDJGbhjT7t8E4pb0Zu49Ludj4hg4wt1_mQ"
```

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)
```

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")
```



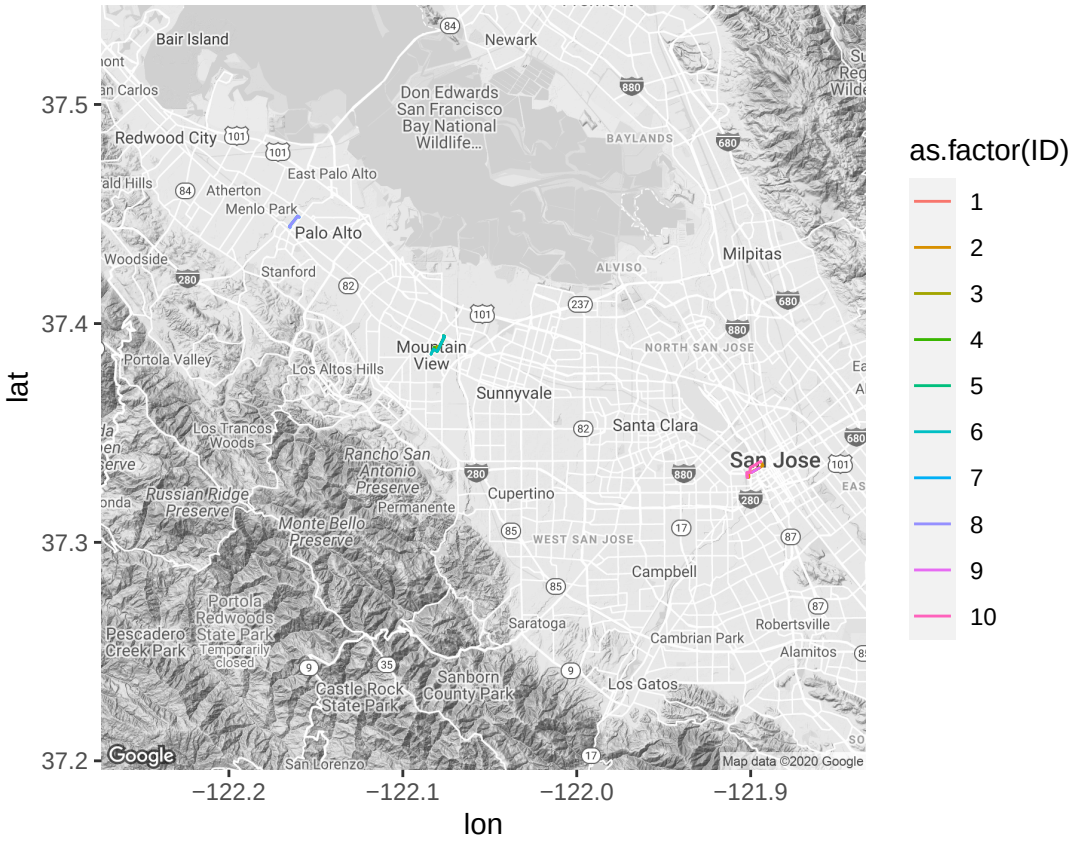
We can extend the above to run a conditional statement with the for() function which does something (in this case, what is in brackets) until a condition is satisfied. Here loop changes the value of x from 1 to the number of rows in the top10 object (i.e. 10), and for each change in x the code between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



As we can see, the top ten most frequented trips are taken in all three cities, not just San Jose.

Exploring Silicon Valley Bikeshare Traffic Flows

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:

```
# Create a list of all flows greater than 0 and within Silicon Valley
ALL <- OD_trips_SV_Narrow[(OD_trips_SV_Narrow$Origin %in% s_SV) | (OD_trips_SV_Narrow$Destination %in% s_SV
ALL <- ALL[ALL$Trips > 0,]
ALL <- ALL[ALL$Origin != ALL$Destination,]
ALL$ID <- 1:nrow(ALL)

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


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


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

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

# Get origins and destinations
origin <- c(ALL[x,"O_lat"],ALL[x,"O_long"])
destination <- c(ALL[x,"D_lat"],ALL[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$ID <- ALL[x,"ID"]
df_polyline$Trips <- ALL[x,"Trips"]

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

Sys.sleep(time = 1)
print(x)

}

All_Flows <- tmp_all

save(All_Flows,file="All_Flows.Rdata")
```

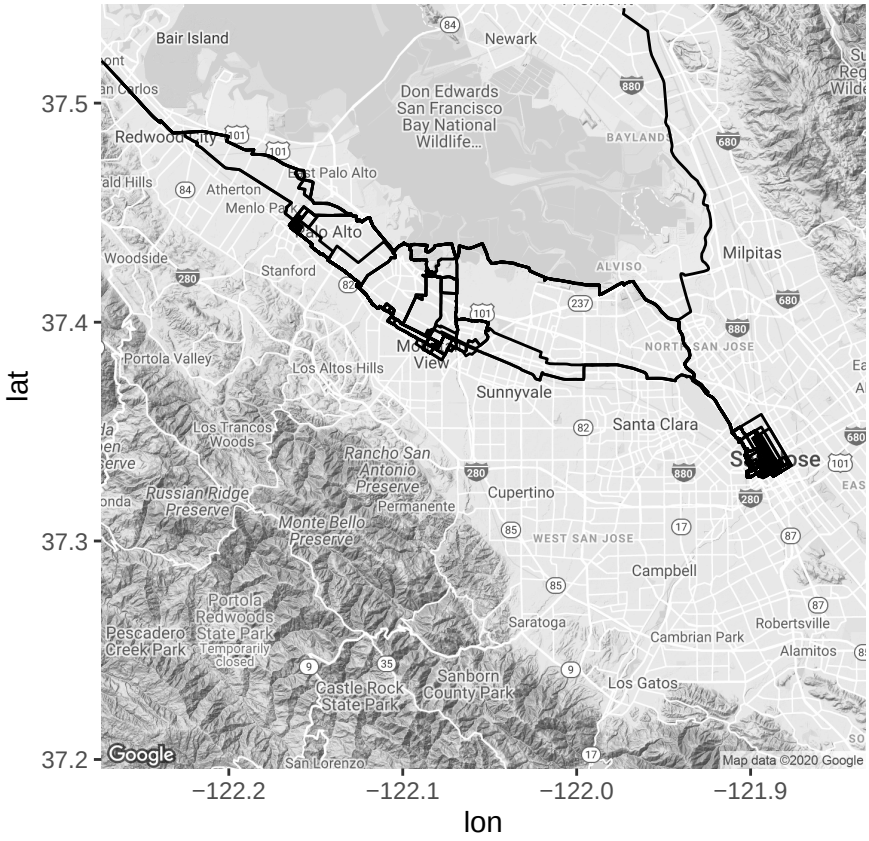
There are in total 446 pairs of stations that have routes to be generated, and we generate routes between each of them and collect trip data between these station pairs.

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

Visualizing the trips

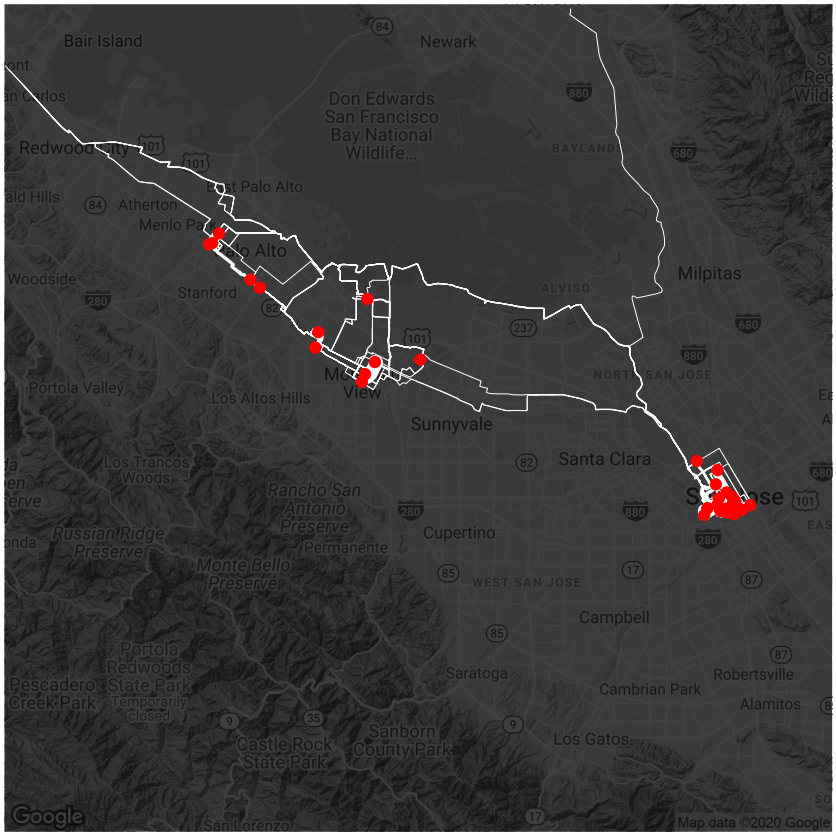
We can now show these on a map - we use the group option within the aes to tell ggmap that these are id that separate the routes, otherwise the whole set of co-ordinates are interpreted as a single route. We can remove these and generate the plot again to see what happens.

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



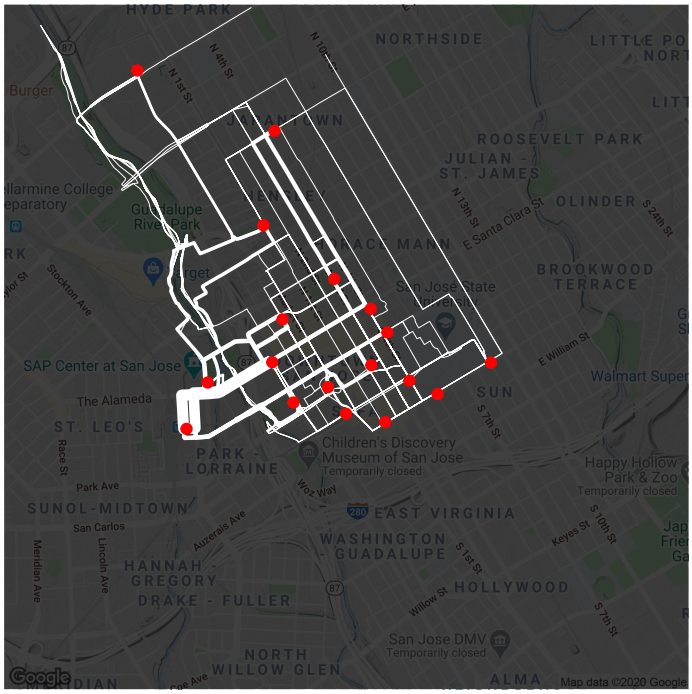
We can now use the trip information to adjust the plot - for example, to scale the routes by the flow volume. We add the size option, but also divide the flows by 1000 to make the line widths an acceptable size. Thicker lines represent greater flows. We have also added the location of the stations in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

```
ggmap(valley,darken = 0.75) +
  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") +
  theme (
    axis.text = element_blank (),
    axis.title = element_blank (),
    axis.ticks = element_blank ()
    )
```

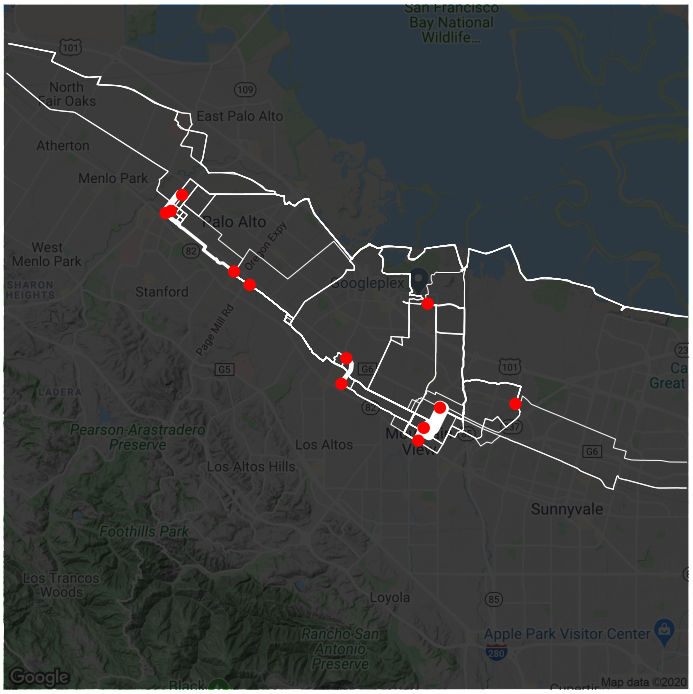


Visualizing this in the two city-level clusters (code not shown for asthetic purposes, and the code is similar to before):

San Jose

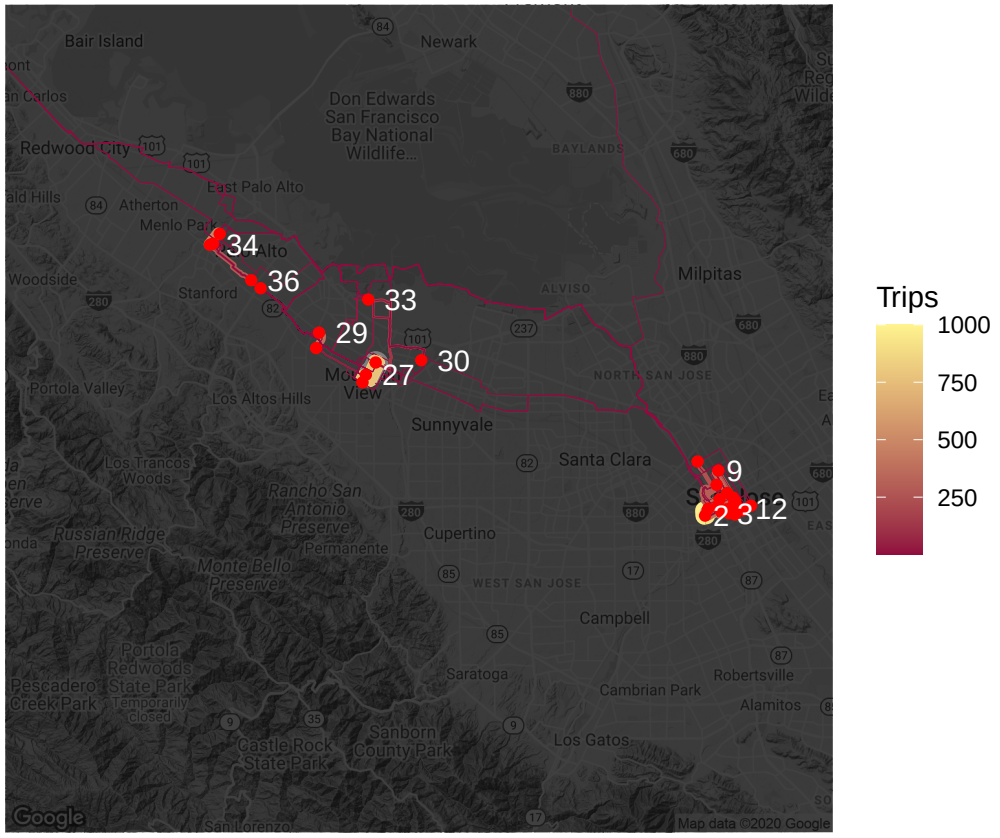


Palo Alto + Mountain View



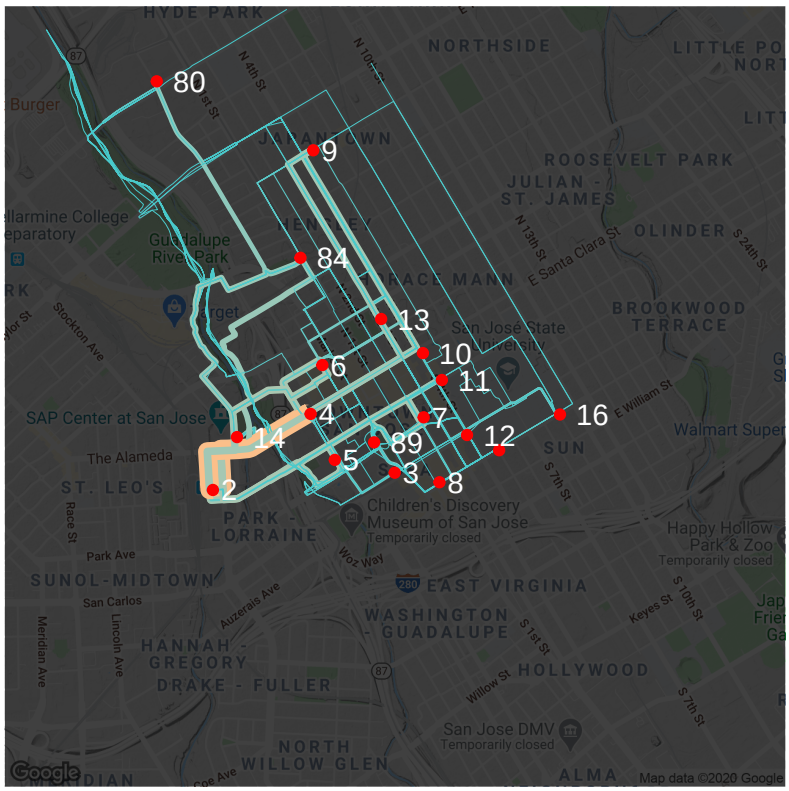
Alternatively, we can achieve this by coloring the lines by intensity of flow; plus, we have also added some labels for the station ID using geom_text():

```
ggmap(valley,darken = 0.75) +
  geom_path(aes(x = lon, y = lat, group = ID,colour = All_Flows$Trips), data = All_Flows, size = All_Flows$
  scale_colour_gradient(limits = c(1,1000), low="#900C3F", high="#FFF48E",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 ()
        )
```

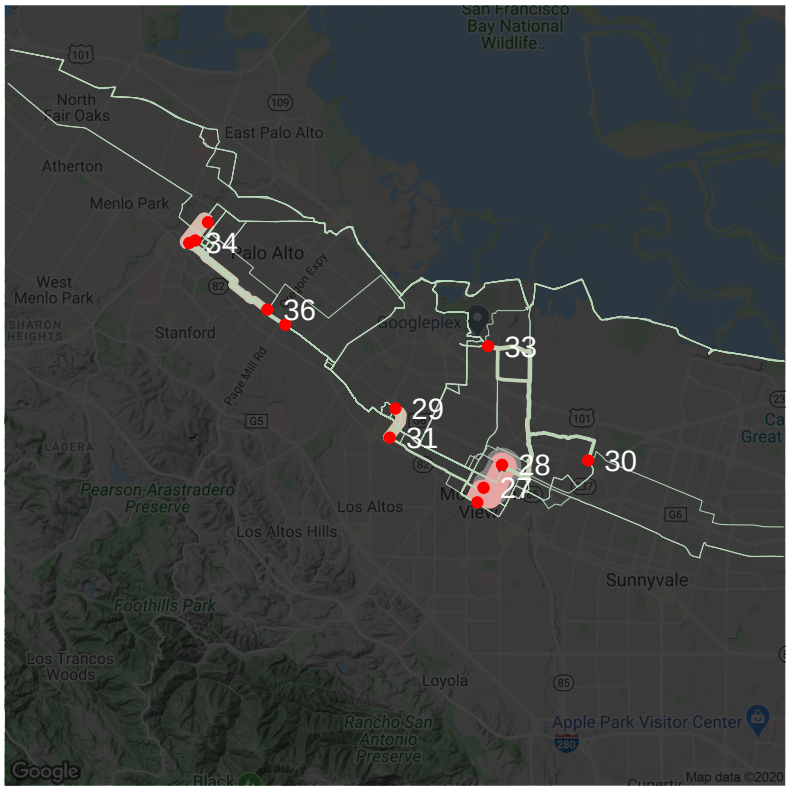


Again, visualizing in the city-level:

San Jose



Palo Alto + Mountain View



Conclusion

From the above traffic flow analyses, we can conclude that outside of San Francisco, for the rest of the Bay Area, the bikesharing program is also interesting for analyses. There are two city-level clusters: San Jose, Mountain View + Palo Alto. The trips are mostly within the clusters and the bike stations are located in the centers of the city, and near major tech company campuses. Traffic flows are also the most significant near tech comapnies and on main boulevards. Due to the more flat nature in south bay, the bike routes are actually more direct between stations.