Objectives

Transit Lab

Sylvia de Boer

Objectives

I took the first case presented in the Transit Lab, which involved the mapping of individual bike routes in San Francisco, and chose to investigate some different variables in the hopes of organizing routes differently. It didn't turn out exactly as I wanted, but it was still a helpful exercise in pulling and manipulating the data.

Data Loading

So first, I read in datasets for stations and trips. I limited the stations to those in San Francisco.

```
stations <- read.csv("/Users/sylviadeboer/Desktop/201608_station_data.cs
v")
trips <- read.csv("/Users/sylviadeboer/Desktop/201608_trip_data.csv")
head(stations)
```

```
##   station_id                               name      lat      long dock
count
## 1          2 San Jose Diridon Caltrain Station 37.32973 -121.9018
27
## 2          3             San Jose Civic Center 37.33070 -121.8890
15
## 3          4            Santa Clara at Almaden 37.33399 -121.8949
11
## 4          5                 Adobe on Almaden 37.33141 -121.8932
19
## 5          6                 San Pedro Square 37.33672 -121.8941
15
## 6          7              Paseo de San Antonio 37.33380 -121.8869
15
##   landmark installation
## 1 San Jose     8/6/2013
## 2 San Jose     8/5/2013
## 3 San Jose     8/6/2013
## 4 San Jose     8/5/2013
## 5 San Jose     8/7/2013
## 6 San Jose     8/7/2013
```

```
stations <- stations[stations$landmark == "San Francisco",]
```

Downloading packages

So next, I refreshed ggmap and dplyr. GGMAP is a mapping package, while dplyr allows us to select, subset, and otherwise manipulate spatial data.

```
library(ggmap)
```

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

```
## Google's Terms of Service: https://cloud.google.com/maps-platform/terms/.
```

```
## Please cite ggmap if you use it! See citation("ggmap") for details.
```

```
library(dplyr)
```

```
##
## 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
```

Registering API key

Next, I registered my Google API, and make sure in my Google console that Static Places API was enabled (I ran into problems with that last time I used ggmap).

```
register_google(key="AIzaSyDUAyb7mbuI6hK3EfL9v3WE1fwfOsnsaaA")
key <- "AIzaSyDUAyb7mbuI6hK3EfL9v3WE1fwfOsnsaaA"
```

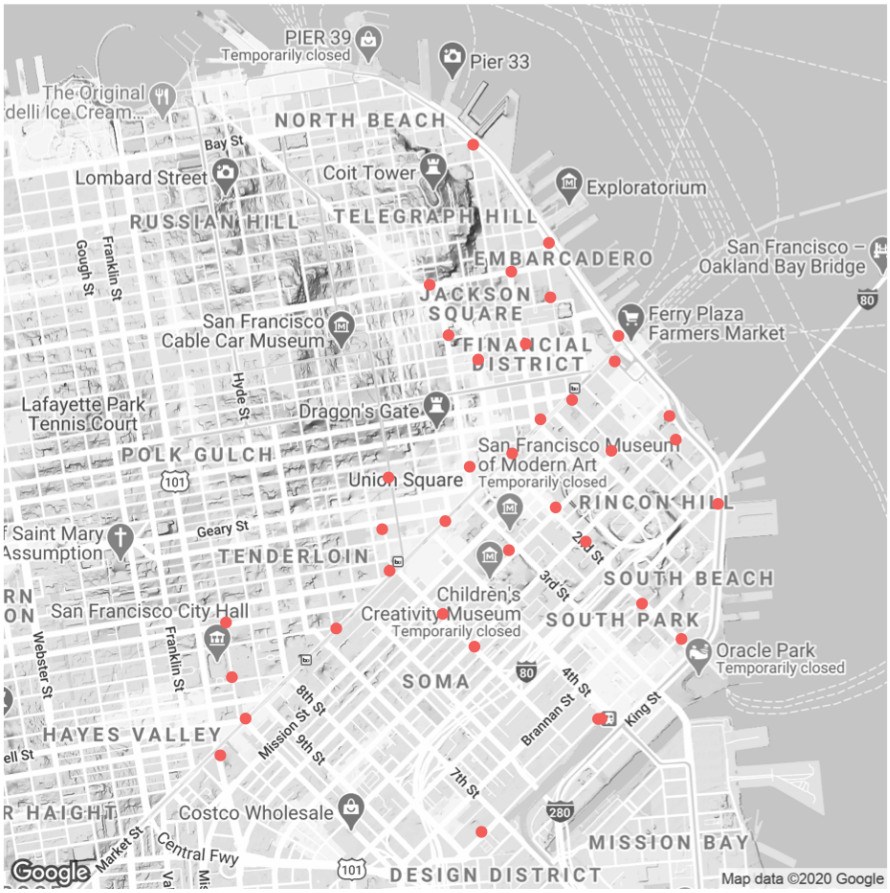
Setting up map

So next, we pull a map of San Francisco. Then we map stations onto the map, using ggmap. The package makes it intuitive to layer elements on top of each other with the +.

```
SanFran <- get_map(location = c(-122.405,37.79), zoom = 14,color = 'bw')
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=37.79,-122.405&zoom=14&size=640x640&scale=2&maptype=terrain&language=en-EN&key=xxx
```

```
ggmap(SanFran) + geom_point(data = stations, aes(x = long, y = lat, colour = "red")) +
  theme_bw() +
  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())
```



Defining our initial DB of San Fran trips

So first, we'll pull a list of station IDs that are present in San Francisco.

```
s_SF <- unique(stations$station_id)
```

Next, we'll limit our results to trips with origins and destinations in San Francisco.

```
trips_SF <- trips[(trips$Start.Terminal %in% s_SF) & (trips$End.Terminal
 %in% s_SF),]
```

Narrowing down to City Hall trips

Next I created a table that includes the start terminal (origin), end terminal (destination), and end station. I chose to include end station because the end stations are indicated as places in the dataset, and I wanted to focus on trips to San Francisco City Hall.

```
OD_trips_SF1 <- table(trips$Start.Terminal,trips$End.Terminal,trips$End.S
tation)
```

Next, I created a data frame of the OD pairs.

```
OD_trips_SF_Narrow1 <- data.frame(OD_trips_SF1)
```

Then I subsetted the data to include only trips with a final station of City Hall. At this point, end stations were in the dataset under the columnn "Var3", so I specified that using subset, a tool within dplyr.

```
cityhall <- subset(OD_trips_SF_Narrow1, Var3=="San Francisco City Hall")
```

Next, I added origin and destination latitude and longitude co-ordinates by merging with the stations data.

```
cityhall <- merge(cityhall,stations, by.x="Var3",by.y="station_id", all.x
=TRUE)
```

I removed unwanted columns.

```
cityhall <- subset(cityhall, select=c("Var1","Var2","Var3","lat","long"))
```

I changed the names of the columns, probably belatedly but that's okay:

```
colnames(cityhall) <- c("Origin","Destination","EndStation","O_lat","O_lo
ng")
```

I added destination coordinates by merging the station dataset, which includes the lat/long of each station, with our original dataset.

```
cityhall <- merge(cityhall,stations, by.x="EndStation",by.y="station_id",
all.x=TRUE)
```

So then I refreshed googleway, which is useful for calculating directions.

```
library(googleway)
```

So then I randomly pulled an origin-destination pair from our set of cityhall trips, and then use the google_directions() function to generate a route.

```
x <- 2
origin <- c(cityhall[x,"O_lat"],cityhall[x,"O_long"])
destination <- c(cityhall[x,"O_lat"],cityhall[x,"O_long"])
```

Then I used the google_directions tool to calculate the directions from the origin to the destination that we just pulled. I decided to use the walking mode.

```
res3 <- google_directions(origin = origin,destination = destination, key
= key, mode= "walking")
```

Then I used decode_pl to convert the route to coordinates.

```
df_polyline3 <- decode_pl(res3$routes$overview_polyline$points)
```

And finally, I mapped (or attempted to map) the route on top of the San Francisco base map. It turned out to be a revelation just to get it to load without an error - in many previous iterations I got the error message that “each group consists of only one observation”. I honestly do not know what I did differently, I just reloaded everything and got it to (sort of) work.

```
ggmap(SanFran) +
geom_path(aes(x=lon, y=lat), colour = "red", size = 0.5, data = df_polyli
ne3, lineend = "round")
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

So once I did this, I realized that the map wasn't showing anything (obviously) so I checked with the route I had drawn and realized that it was a very, very, very short route. I'm not really sure why this happened - I tried other iterations of x, and still got almost nonexistent routes.

```
res3
```

```
## $geocoded_waypoints
##   geocoder_status                    place_id              types
## 1              OK ChIJCzYy5IS16lQRQrfeQ5K5Oxw country, political
## 2              OK ChIJCzYy5IS16lQRQrfeQ5K5Oxw country, political
##
## $routes
##   bounds.northeast.lat bounds.northeast.lng bounds.southwest.lat
## 1             37.08975            -95.71249             37.08975
##   bounds.southwest.lng     copyrights
## 1            -95.71249 Map data ©2020
##
legs
## 1 1 m, 0, 1 min, 0, United States, 37.0897452, -95.7124868, United Sta
tes, 37.0897452, -95.7124868, 1 m, 0, 1 min, 0, 37.0897452, -95.7124868,
Head on <b>Estate Enighed</b><div style="font-size:0.9em">Restricted usag
e road</div>, }akaF`zdgQ, 37.0897452, -95.7124868, WALKING
##       points        summary
## 1 }akaF`zdgQ Estate Enighed
##
warnings
## 1 Walking directions are in beta. Use caution – This route may be miss
ing sidewalks or pedestrian paths.
##   waypoint_order
## 1           NULL
##
## $status
## [1] "OK"
```