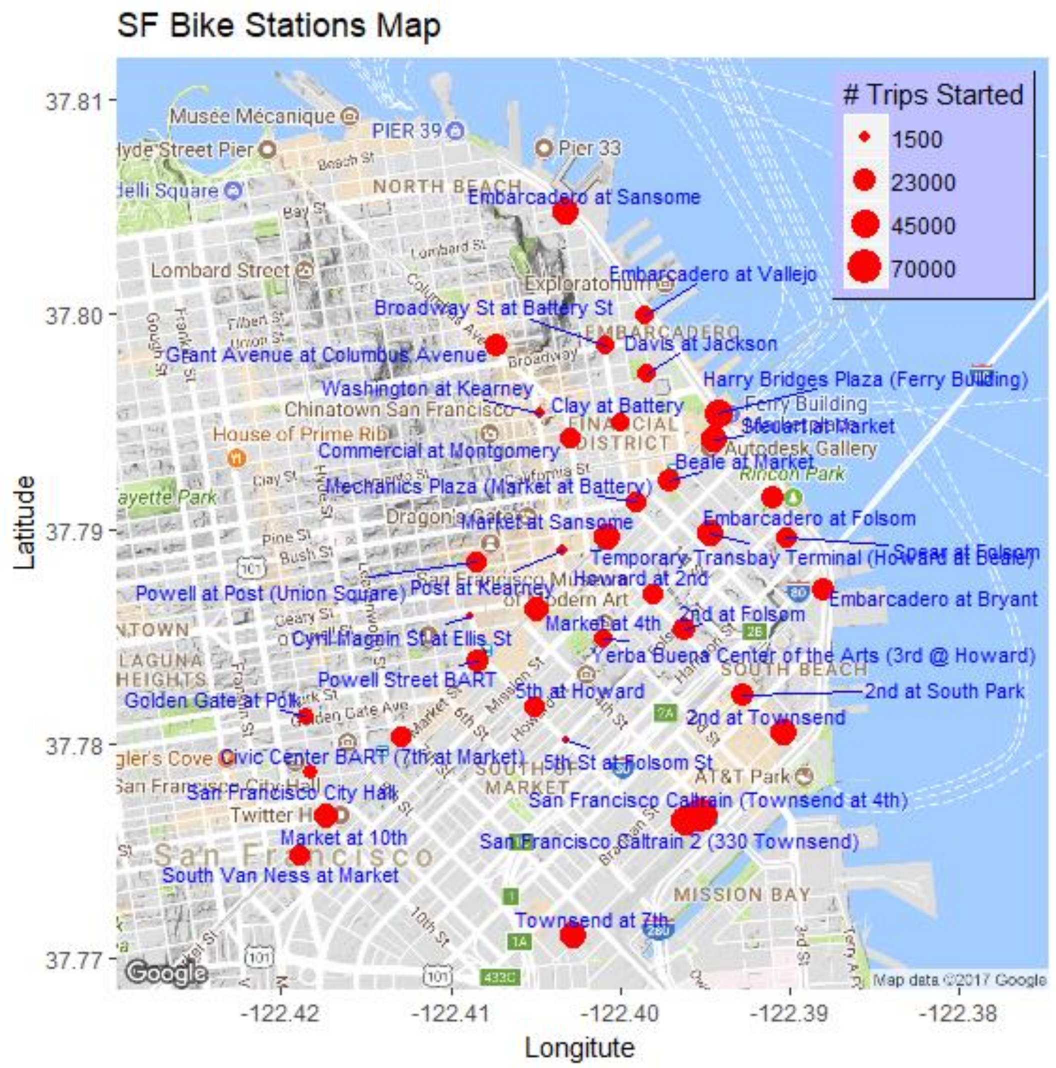
Mitchell Layton

912307956

STA 141A: Homework Assignment # 3 – due Tues, November 14th 5:00 PM

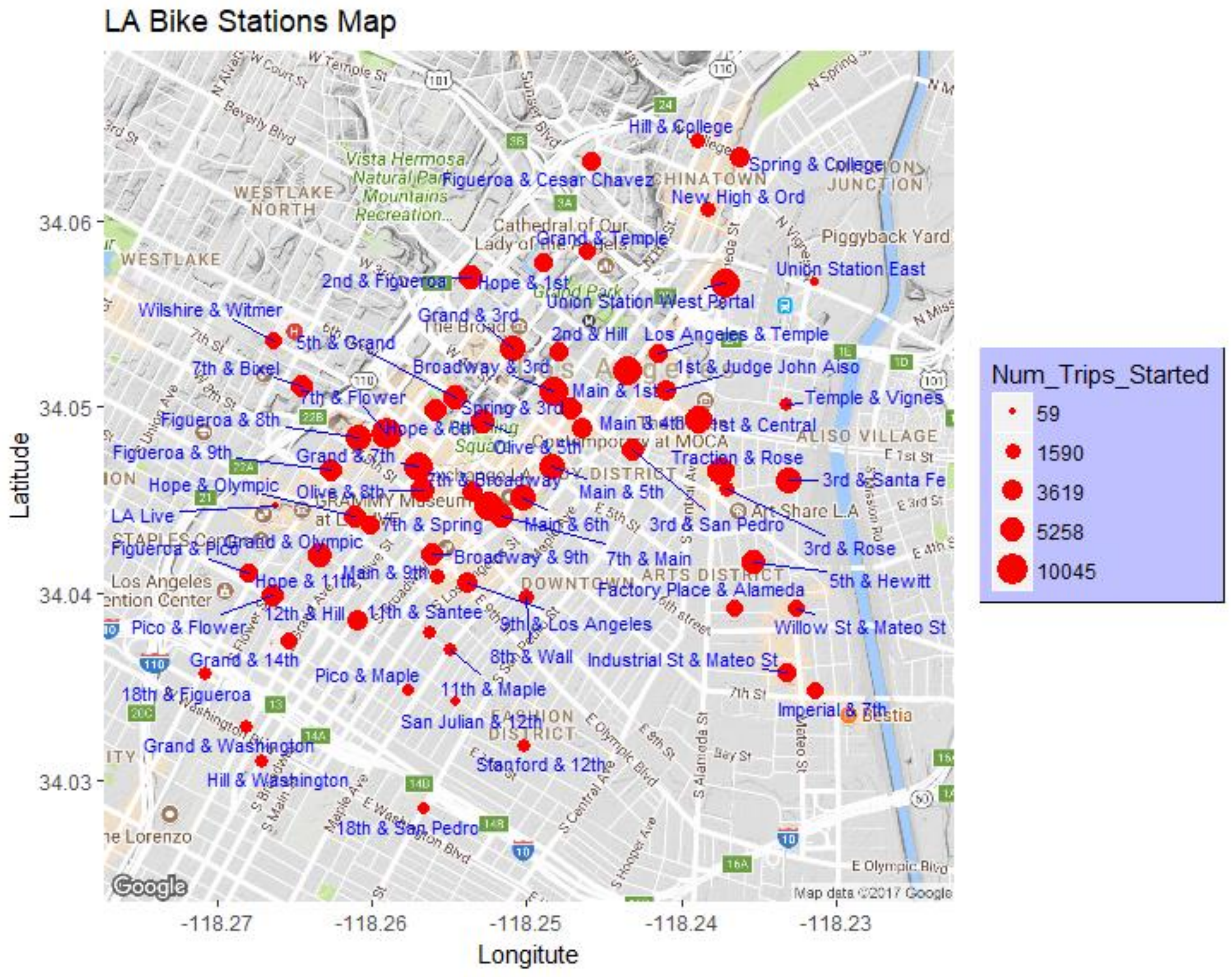
1) (code in appendix)



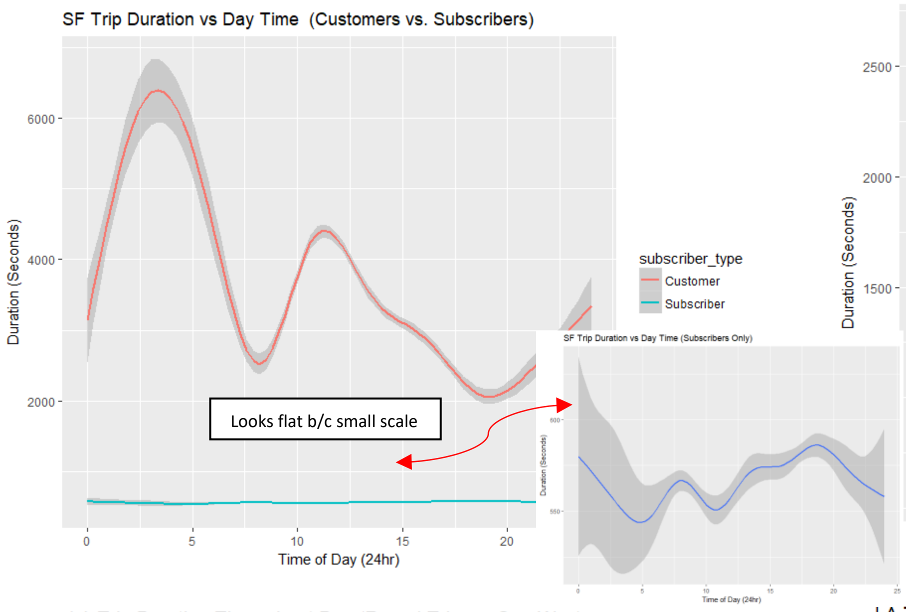
2) The graphic above displays a google map of the northeast bay of the San Francisco and the accompanying bike share stations with their respective names. The red dots represent the locations of the stations and are size dependent upon the number of trips started from the stations. Some observations we notice here are there are some clear front runners in the amount of trips started from that station. It may be a result of location or foot traffic such as the case for the max value of the trips started data which is the San Francisco Caltrain (Townsend at 4th) which is near the lower right side of the bay almost in between Mission Bay and AT&T Park.

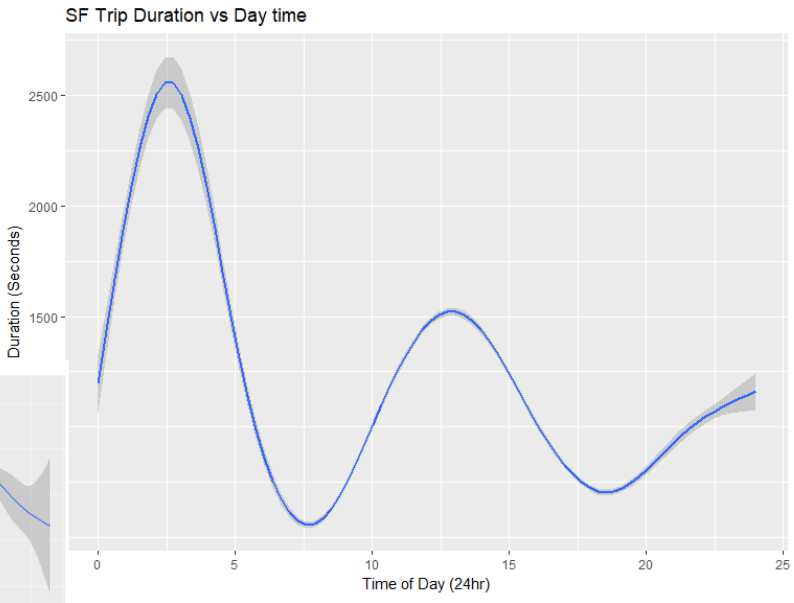
From the time of this data, this place had 72,683 trips started which is huge compared to the average trips started of 23,588. Another observation of the data points on the map is that they all usually tend to the middle right side of the SF peninsula probably because of the great tourist spots and places to visit on that side. Additionally, starting from the bottom left at South Van Ness at Market all the way up to Harry Bridges Plaza (Ferry Building) there is a line of bikeshare stations that run all along Market Street which seems to be the most popular and best place to have a bikeshare station.

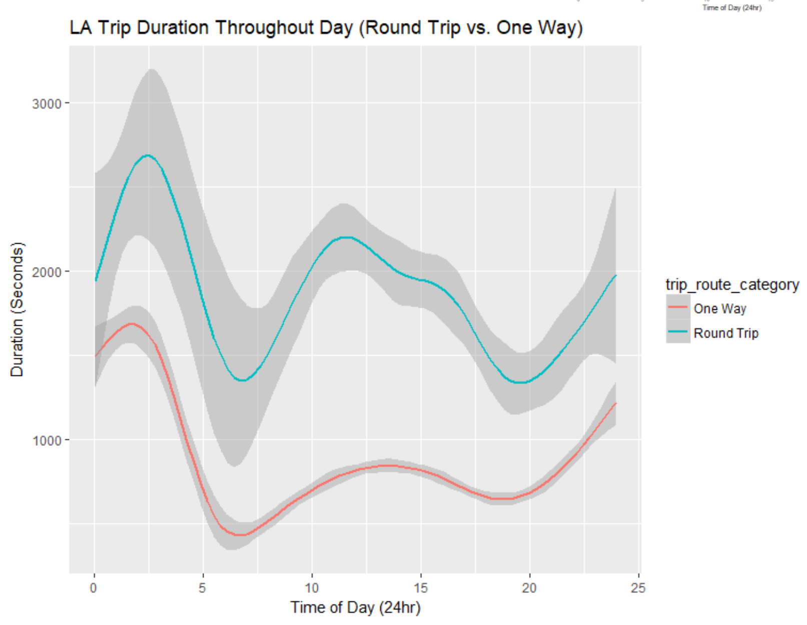
3) (code in appendix)

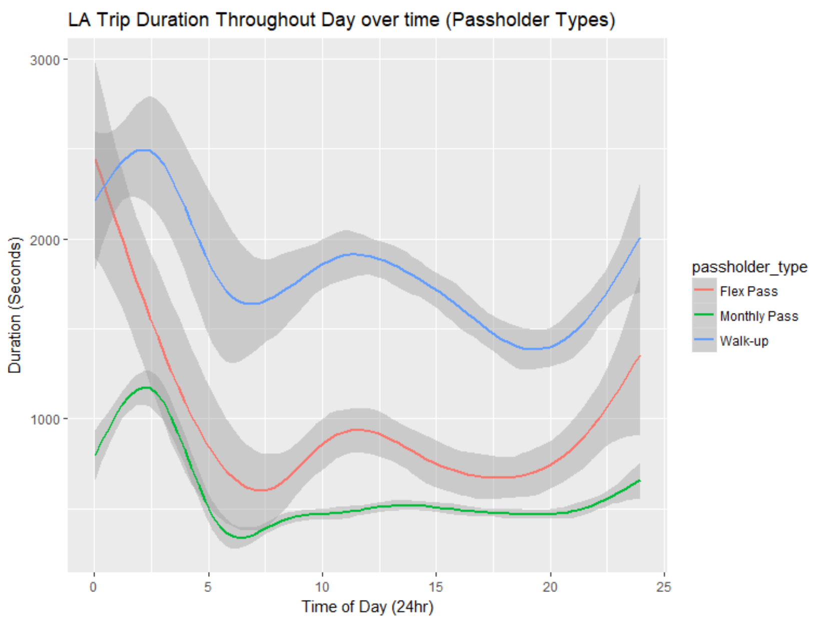


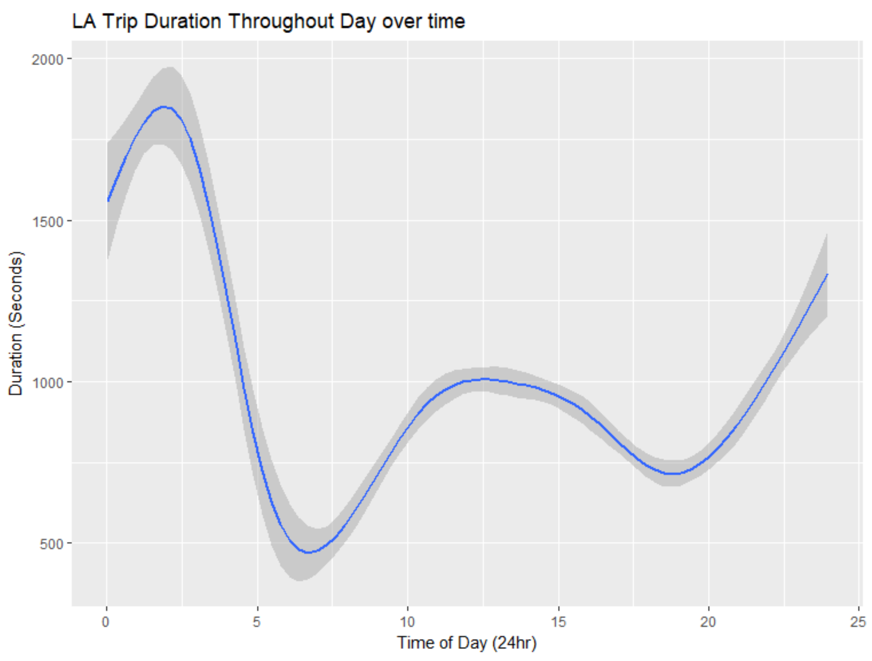
4) Above is the map of Downtown LA and the accompanying bike stations plotted based on the number of trips started from that station. Looking at the plot, one may notice that the middle most area has the bunch of most of the medium to big bike stations with the smaller ones being pushed towards the outsides. The bigger point aggregating towards the middle must be where the heart of downtown LA is and include many tourist locations suitable for biking distances. The spread of the bike stations is decent with not too many clumps of close data points. Really the major spread of the data is in an ellipse slanted upwards at an angle. For the number of trips in the legend I used the summary function to find the best sized breaks for the data points.



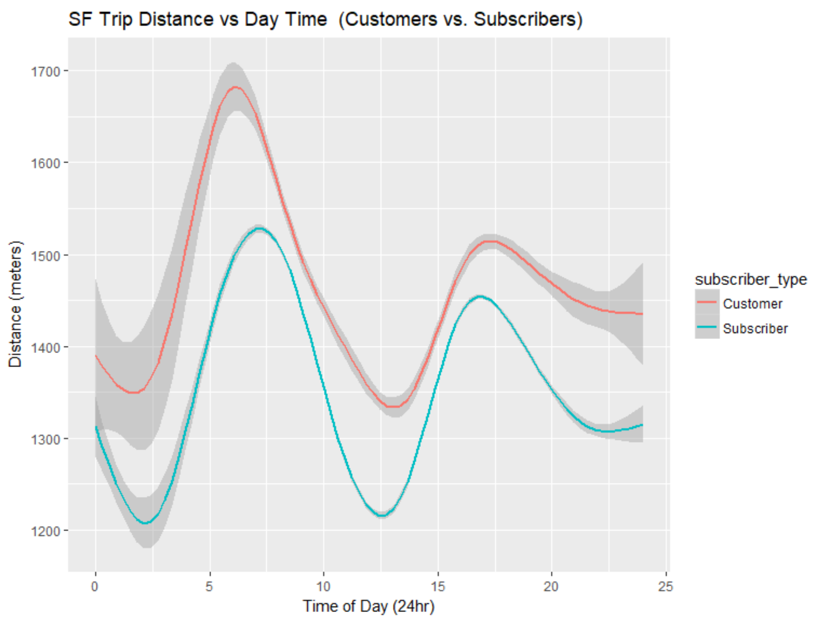


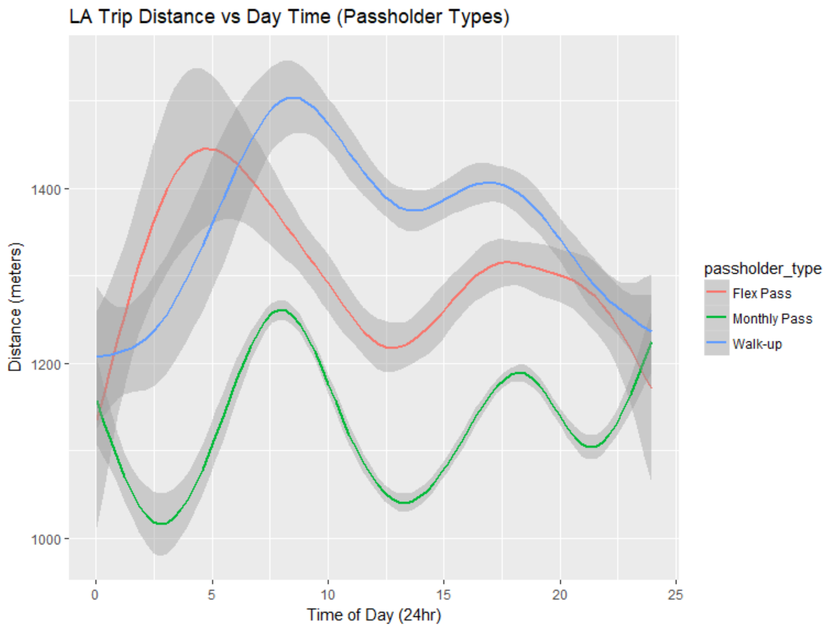


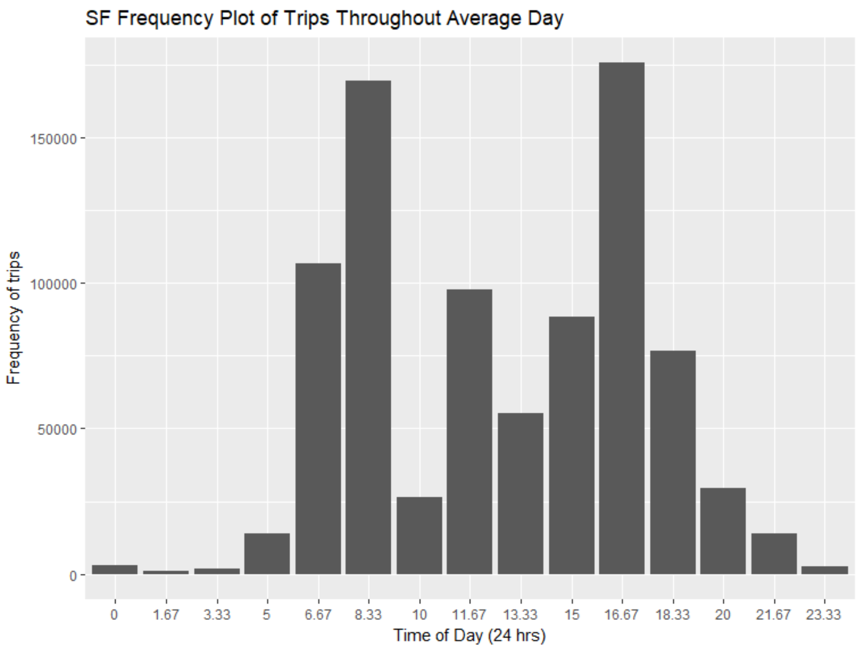




5) First I subset the data on durations of trips less than 24 hrs. because there were clear outliers that were messing up the plots. While also using na.omit to account for any missing data points. This question was the one that required most of the set up in the code. I started off by finding how duration of the trips changed at different times of the day. After doing the appropriate joins and sub-setting of data I came with the 3 graphs for SF duration times. The first compares customers versus subscribers of the bikeshare stations. We notice the customers cover a much larger range of time. Furthermore, we notice the large peak between 12:00 AM – 5:00AM which leads me



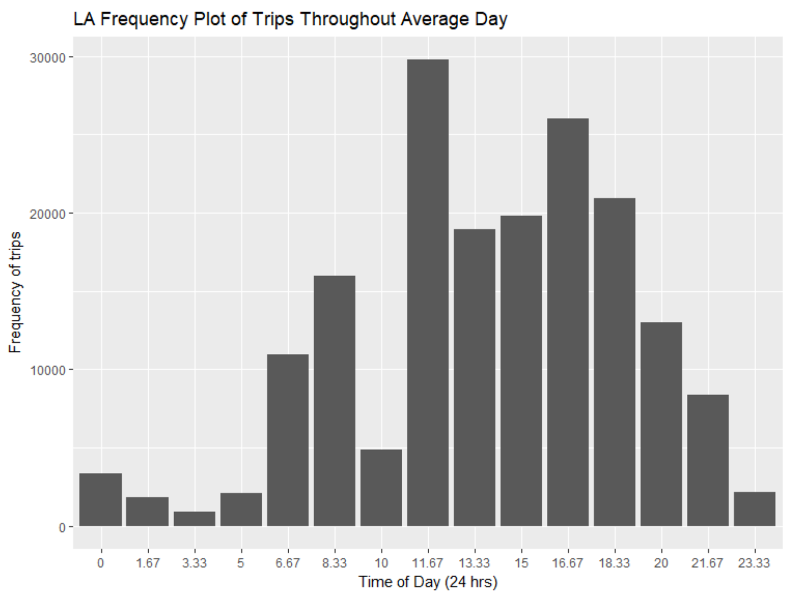




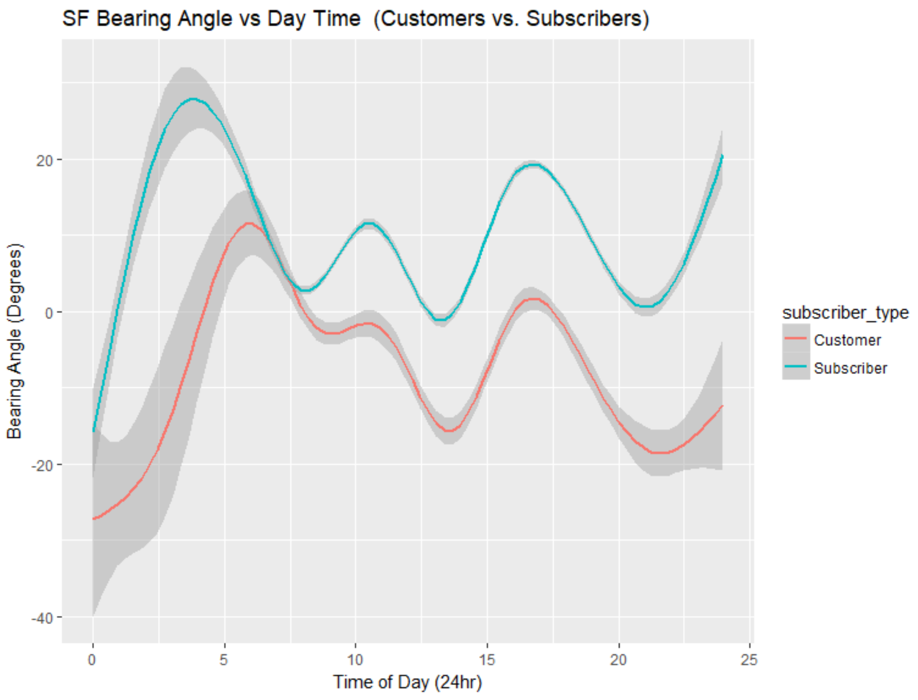
to infer that many customers like returning their bikes after long nights with friends or going out. My inference of why subscriber's duration times are so low is that those who opt into a subscriber service may use it as an alternative to public transportation or convenience of personal transportation needs and that is why those people are not going on super long bike rides. The other obvious peak from the top right graph of the previous page is when bikes are being returned during the midafternoon times.

While the Downtown LA data had more metrics to compare the target data variable to, we notice similar trends in the trends of the slopes. For one the LA graph (Road Trip vs One-Way) it is obvious that the one-way durations will be much shorter but the graph to its right with the different passholder types show some very interesting things. The walk-ups in general always have the highest trip duration, while the monthly pass users have the lowest. It makes sense for people who have monthly passes to not feel the need to use their service to its fullest each time by taking long trips. That trend I had mentioned about people returning their bikes after long nights from the 0-5-hour mark (24-hr clock) holds true for both the flex and walk-up passholder types, but does not for the flex-pass. The flex path shows a very steep decline during that late night/early morning time frame which could have something to do with the logistics of the flex pass capabilities.

Next, the distance throughout the course of the day tells a little different story between the two cities. Comparatively, SF's plot in general has steeper trend lines as well as smaller error lines throughout the day particularly for



subscribers. People seem to be commuting or traveling at those peak early morning hours around 6:00 AM. From the height of the peaks we notice SF people area generally biking a little further during those morning hours than those people in LA. Distance, especially for subscribers in SF seems to be a constant because we notice barely any error line in our smooth line plot. Subscribers know what they need or want and that is why they opted into buying that bike service and are traveling similar distances each time. Where SF customers' and subscribers' follow a similar looking distribution and spread, LA's different passholders' generate different looking distributions. The monthly users tend to not travel as far and not as early as the other two types. Interestingly, the flex pass users are traveling more much earlier than any of them for some reason. Lastly, I created a histogram of the frequency of trips throughout the day by counting the number of trips for 3 times for the 5 normal time intervals in the other graphs shown. It follows our habitual work and lifestyle patterns consistently with most trips happening around 7:00-8:00 AM and as well in the evening after work around 5:00-6:00 PM. In LA frequency of trips graph the number of trips is much closer together from around noon until 8:00PM. Since the terrain is a little more forgiving in LA, maybe they are using the bikeshare stations more as main a transportation service.



6) An interesting observation of the bearing angles versus time graph is how the concavity shifts from the early U-shaped peaks where the morning workout/travel times around 6:00-8:00 AM are, to the regular slope looking peak around when people get off work. The degree differences in their travel habits can differ up to and sometimes over the 20-degree mark. Furthermore, these bearing angles represent a circular biking motion where say they leave at a 0-degree angle at hour 13, the steadily increase their bearing angle up to the peak where it starts to slope back down at a decreasing angle. We can infer that the peaks are representative of a circular or cyclical biking fashion where one bikes to and from their location. With this graph, we can more easily identify the bikers' traveling behavior times. It's during these peaks where the angles changes that I think traffic starts to become a problem because it tells us these bikers are leaving right back from where they had started the trip which then we can follow logically that it must include automotive traffic as well.

Code Appendix

```
library(tidyverse)
library(ggmosaic)
library(plyr)
library(dplyr)
library(ggthemes)
library(grid)
library(sqldf)
library(reshape)
library(plotly)
library(scales) # to access breaks/formatting functions
library(car)
library(RColorBrewer)
library(rio)
library(readr)
library(ggmap)
library(ggrepel)
library(data.table)
library(lubridate) # for working with dates
library(gridExtra) # for arranging plots
#-----------------------------------------------------------------------------------------------------------------------------------------
-------------
# 1.)
  #FUNCTION 1
  bike_trips = function(path_csv, output_rds, relative_path) {

    # Set path for input CSV file
    x = read_csv(path_csv)
    x$start_date = as.POSIXct(strptime(x$start_date, "%Y-%m-%d %H:%M:%S"))
    x$end_date = as.POSIXct(strptime(x$end_date, "%F %H:%M:%S"))
    x$start_station_id = as.factor(x$start_station_id)
    x$end_station_id = as.factor(x$end_station_id)

    # save our altered columns and data frame into .rds file and read it
    setwd(relative_path)
    y = saveRDS(x, file = output_rds)
    y =  readRDS(output_rds)
```

```
        return(y)
    }
        # (3) parameters, (1) = Direct Path of csv infile for sf bikeshare trips (2) = .rds output file name. (3)
relative path for where to save .rds file
        data_hw3 = bike_trips("E:/Davis/STA 141A/DATA/sf_bikeshare_trips.csv",
"sf_bikeshare_trips.rds","E:/Davis/STA 141A/DATA/")
        data_hw3 = as.data.table(na.omit(data_hw3))
        data_hw3


#FUNCTION 2
    SF_bike_stations = function(path_csv, output_rds, relative_path) {


        # Set path for input CSV file
        x = read_csv(path_csv)
        x$installation_date = as.POSIXct(strptime(x$installation_date, "%Y-%m-%d"))
        x$station_id = as.factor(x$station_id)


        # save our altered columns and data frame into .rds file and read it
        setwd(relative_path)
        y = saveRDS(x, file = output_rds)
        y =  readRDS(output_rds)
        return(y)
    }


        sf_stations = SF_bike_stations("E:/Davis/STA 141A/DATA/sf_bike_share_stations.csv",
"sf_stations.rds", "E:/Davis/STA 141A/DATA/")
        sf_stations


#-------------------------------------------------------------------------------------------------------------------------------------
----
# 2.)
    # Create map with locations of Bay Area bike share stations in SF only
    # CODE REFERENCE FOR MAP: http://eriqande.github.io/rep-res-web/lectures/making-maps-with-
R.html



    # use data.table becuase has powerful funcitonality for wrangling data: https://cran.r-
project.org/web/packages/data.table/data.table.pdf
    sf_stations = filter(sf_stations, landmark == "San Francisco")
    sf_stations = unique(sf_stations)

    sf_stations = as.data.table(sf_stations)

    # Since there are duplicates of the data lets further subset it to ensure only distinct names of the
stations

    sf_stations = sf_stations[(duplicated(sf_stations, by = "name")) == F]


    counts_of_start = filter(data_hw3, start_station_name %in% sf_stations$name)

    counts_of_start = as.data.table(table(counts_of_start$start_station_name))

    names(counts_of_start) = c("name", "# Trips Started")


    # Set the keys of our data.tables so that we can use data.table join functionality

    setkey(counts_of_start, name)

    setkey(sf_stations, name)

    # result variables joins our N variable (the frequency of # of trips started) from counts_of_start table

    result = sf_stations[counts_of_start, nomatch=0]



    # Picked 2 addresses by eye-balling from Google maps to fit into my zoomed map for best visual

    address1 = geocode("900 Market St, San Francisco")

    address2 = geocode("70 Washington St, San Francisco")

    gc = unlist((address1+address2)/2)

    map = get_map(gc, zoom = 14)


    # Using geom_text_repel() to fit text names on map easier: refered from Piazza

    # and from https://cran.r-project.org/web/packages/ggrepel/vignettes/ggrepel.html

    SF_map = ggmap(map) +

        geom_point(data = result, aes(x = longitude, y = latitude, size = `# Trips Started`), color = "red") +

        geom_text_repel(

           data = result

          ,aes(longitude, latitude, label = name)

          ,color = "blue", size = 3.0

        ) +

        scale_size_continuous(breaks=c(1500,17500, 40000, 70000)) +

        labs(x = "Longitute", y = "Latitude", title = "SF Bike Stations Map") +

        theme(

          legend.position = c(.985,.985),

          legend.justification = c("right", "top"),
            legend.box.just = "right",
            legend.margin = margin(4,4,4,4)
        ) +
        theme(
            legend.background = element_rect(fill = "#BEC0FF"),
            legend.box.background = element_rect(fill = "black")
        )
    SF_map

#---------------------------------------------------------------------------------------------------------------------------------------------
----
# 3.)

    #Function 1:  LA Bike Share trip data for 5 CSV files: combining 5 csv files using data.table functions
(rbindlist & fread)

        # Reference: https://stackoverflow.com/questions/30399446/how-to-read-in-multiple-data-tables-
into-r-using-apply-function

   La_trips = function(La_data_dir, output_rds) {

        # La_data_dir = relative path to where the la_metro csv files are stored in
        filenames = list.files(La_data_dir, pattern = "la_metro_trips", full.names = T)

        # Retrieve list of all files names with key words
        result = rbindlist(lapply(filenames, fread))

        # Convert the column types
        result$start_time = as.POSIXct(strptime(result$start_time, "%m/%d/%Y %H:%M"))
        result$end_time = as.POSIXct(strptime(result$end_time, "%m/%d/%Y %H:%M"))
        result$start_station_id = as.factor(result$start_station_id)
        result$end_station_id = as.factor(result$end_station_id)
        result$start_lat = as.double(result$start_lat)
        result$end_lat = as.double(result$end_lat)
        result$start_lon = as.double(result$start_lon)
        result$end_lon = as.double(result$end_lon)

        # save our altered columns and data frame into .rds file and read it
        # using Relatvie path for saving
        setwd(La_data_dir)
        y = saveRDS(result, file = output_rds)
        y =  readRDS(output_rds)
        return(y)

    }

        metro_data = La_trips("E:/Davis/STA 141A/DATA/","la_bikeshare_trips.rds")   # Enter both
parameters for function, 1) The relative path & 2) The output rds file
        metro_data = as.data.table(na.omit(metro_data))
        metro_data




    # FUNCTION 2: LA bike share stations data
    LA_bike_stations = function(path_csv, output_rds, relative_path) {

        # Set path for input CSV file
        x = read_csv(path_csv)
        x$Go_live_date = as.POSIXct(strptime(x$Go_live_date, "%m/%d/%Y"))
        x$Station_ID = as.factor(x$Station_ID)

        # save our altered columns and data frame into .rds file and read it
        setwd(relative_path)
        y = saveRDS(x, file = output_rds)
        y =  readRDS(output_rds)
        return(y)
    }

        la_stations = LA_bike_stations("E:/Davis/STA 141A/DATA/metro-bike-share-stations-2017-10-
20.csv", "la_stations.rds", "E:/Davis/STA 141A/DATA/")
        la_stations

#-----------------------------------------------------------------------------------------------------------------------------------
----
# 4.)
        # Create map with locations of LA's bike share stations near Downtown LA only
        # CODE REFERENCE FOR MAP: http://eriqande.github.io/rep-res-web/lectures/making-maps-with-
R.html



        # use data.table becuase has powerful funcitonality for wrangling data: https://cran.r-
project.org/web/packages/data.table/data.table.pdf
        la_stations = filter(la_stations, Region == "DTLA")
        la_stations = unique(la_stations)
        la_stations = as.data.table(la_stations)
        # To account for any potential duplicated data messing with data manipulation
        la_stations = la_stations[(duplicated(la_stations, by = "Station_ID")) == F]

        counts_of_start_2 = filter(metro_data, start_station_id %in% la_stations$Station_ID)
        counts_of_start_2 = as.data.table(counts_of_start_2)

        new_start_counts = as.data.table(table(counts_of_start_2$start_station_id))
        new_start_counts = new_start_counts[new_start_counts$N != 0]
        names(new_start_counts) = c("Station_ID", "Num_Trips_Started")

        # Set the keys of our data.tables so that we can use data.table join functionality
        setkey(la_stations, Station_ID)
        setkey(new_start_counts, Station_ID)
        # result variables joins our N variable (the frequency of # of trips started) from counts_of_start table
        result2 = la_stations[new_start_counts, nomatch=0]
        setkey(counts_of_start_2, start_station_id)
        result2 = counts_of_start_2[result2, nomatch=0]
        result2 = as.data.table(sqldf("SELECT DISTINCT start_station_id, start_lat, start_lon, Station_Name,
Num_Trips_Started FROM result2"))
        result2 = result2[(duplicated(result2, by = "Station_Name")) == F]



        # Picked 2 addresses by eye-balling from Google maps to fit into my zoomed map for best visual
        la_address1 = geocode("840 Yale St, Los Angeles, CA")
        la_address2 = geocode("508 E Washington Blvd, Los Angeles, CA")
        gc = unlist((la_address1+la_address2)/2)
        map = get_map(gc, zoom = 14)

        # Using geom_text_repel() to fit text names on map easier: refered from Piazza
```

```
        # and from https://cran.r-project.org/web/packages/ggrepel/vignettes/ggrepel.html
        DTLA_map = ggmap(map) +
          geom_point(data = result2, aes(x = start_lon, y = start_lat, size = Num_Trips_Started), color =
"red") +
          geom_text_repel(
            data = result2
            ,aes(start_lon, start_lat, label = Station_Name)
            ,color = "blue", size = 3.0
          ) +
          scale_size_continuous(breaks=c(1250, 3000, 7000)) +
          labs(x = "Longitute", y = "Latitude", title = "LA Bike Stations Map") +
            theme(
              legend.background = element_rect(fill = "#BEC0FF"),
              legend.box.background = element_rect(fill = "black")
            )







            DTLA_map


#-------------------------------------------------------------------------------------------------------------------------------------------
----
# 5.)
        # How do [trip frequency], [distance], and [duration] change at different times of day?


        sf_data = as.data.table(na.omit(data_hw3))
        metro_data = as.data.table(na.omit(metro_data))



        sf_ts = sqldf("SELECT duration_sec, start_date, subscriber_type FROM sf_data WHERE (duration_sec
< 86400) ORDER BY start_date ASC")
        sf_ts = as.data.table(na.omit(sf_ts))
        # lubridate package for extracting the times of the day
        sf_ts$start_date = ymd_hms(sf_ts$start_date)
        sf_ts$start_date = hour(sf_ts$start_date) + minute(sf_ts$start_date)/60


        la_ts = sqldf("SELECT duration, start_time, trip_route_category, passholder_type FROM metro_data
WHERE (duration < 86400) ORDER BY start_time ASC")
        la_ts = as.data.table(na.omit(la_ts))
        # lubridate package for extracting the times of the day
        la_ts$start_time = ymd_hms(la_ts$start_time)
        la_ts$start_time = hour(la_ts$start_time) + minute(la_ts$start_time)/60


# (pt 1) Duration vs time-------------------\


    #SF
        xxx = data.table(sqldf("SELECT * FROM sf_ts WHERE subscriber_type LIKE 'Subscriber' ORDER BY
duration_sec DESC"))


        dur_sf_1 = ggplot(data=sf_ts, aes(x=start_date, y=duration_sec)) +
           geom_smooth(aes(colour = subscriber_type)) +
              labs(x = "Time of Day (24hr)", y = "Duration (Seconds)", title = "SF Trip Duration vs Day Time
(Customers vs. Subscribers)")


        # Made a subscriber plot because scale so small on main dur_sf_1
        subsc = ggplot(data=xxx) +
           geom_smooth(aes(x=start_date, y=duration_sec)) +
           labs(x = "Time of Day (24hr)", y = "Duration (Seconds)", title = "SF Trip Duration vs Day Time
(Subscribers Only)")



        dur_sf_2 = ggplot(data=sf_ts, aes(x=start_date, y=duration_sec)) +
        geom_smooth(na.rm = T)+
           labs(x = "Time of Day (24hr)", y = "Duration (Seconds)", title = "SF Trip Duration vs Day time")


        dur_sf_1
        subsc
       dur_sf_2
    #LA
         dur_la_1 = ggplot(data=la_ts, aes(x=start_time, y=duration)) +
            geom_smooth(aes(colour = trip_route_category)) +
          labs(x = "Time of Day (24hr)", y = "Duration (Seconds)", title = "LA Trip Duration Throughout Day
(Round Trip vs. One Way)")


        dur_la_2 = ggplot(data=la_ts, aes(x=start_time, y=duration)) +
          geom_smooth(na.rm = T)+
          labs(x = "Time of Day (24hr)", y = "Duration (Seconds)", title = "LA Trip Duration Throughout Day
over time")


        dur_la_3 = ggplot(data=la_ts, aes(x=start_time, y=duration)) +
          geom_smooth(aes(colour = passholder_type)) +
          labs(x = "Time of Day (24hr)", y = "Duration (Seconds)", title = "LA Trip Duration Throughout Day
over time (Passholder Types)")


        dur_la_1
        dur_la_2
        dur_la_3


# (pt 2) Distance vs time------------------------\
    library(geosphere)
    #------------------------------------------------------------------------------------------------------------------------------------------
-----------------
    # Pre-plot work for SF


        sf_data = data.table(sqldf("SELECT * FROM sf_data WHERE (start_station_id != end_station_id) &
((start_station_id >= 1) & (end_station_id >= 1))"))
        # need to create a column that gives us the distances into sf_data
        names(sf_stations) = c("station_id","name", "start_lat","start_lon",
"dockcount","landmark","installation_date")


        setkey(sf_stations, station_id)
        setkey(sf_data, start_station_id)
        temp1 = sf_stations[sf_data, nomatch=0]
        setkey(sf_stations, station_id)
        setkey(temp1, end_station_id)
        temp2 = sf_stations[temp1, nomatch=0]
        # Not the prettiest way to combine the data esepcially with confusing variable names but the joins
worked well using data.table "[]" functionality
        Combined_lats_longs = as.data.table(sqldf("SELECT subscriber_type, [i.name] as
start_station_name, [i.start_lon] as start_lon, [i.start_lat] as start_lat, name as end_station_name,
start_lon as end_lon, start_lat as end_lat FROM temp2"))
        Combined_lats_longs$start_date = temp1$start_date[1:862320]
        matri1 = as.matrix(as.data.table(sqldf("SELECT start_lon, start_lat, end_lon, end_lat FROM
Combined_lats_longs")))


        # Subset arguements for distGeo where p1 is first 2 columns of matrix and p2 is last 2 columns for
the latitudes and longitudes
        p1 = matri1[,c(1,2)]
        p2 = matri1[,c(3,4)]


        distance1 = as.data.table(distGeo(p1, p2))
        names(distance1) = "distance"
        Combined_lats_longs$distance = distance1


        # Lastly get hours time like before
        Combined_lats_longs$start_date = ymd_hms(Combined_lats_longs$start_date)
        Combined_lats_longs$start_date = hour(Combined_lats_longs$start_date) +
minute(Combined_lats_longs$start_date)/60
    #-------------------------------------------------------------------------------------------------------------------
    # Pre-plot work for LA -- will be much less work that the data wrangling for SF because metro_data
already includes the
    # starting and ending longitudes and latitudes in main data set along with the times.


        la_dist_data = as.data.table(na.omit(sqldf("SELECT * FROM metro_data WHERE (duration < 86400)
& (start_station_id != end_station_id) & ((start_station_id >= 1) & (end_station_id >= 1))")))


        matri2 = as.matrix(as.data.table(sqldf("SELECT start_lon, start_lat, end_lon, end_lat FROM
la_dist_data")))
        p3 = matri2[,c(1,2)]
        p4 = matri2[,c(3,4)]
        distance2 = as.data.table(distGeo(p3, p4))
        names(distance2) = "distance"
        la_dist_data$distance = distance2


        la_dist_data$start_time = ymd_hms(la_dist_data$start_time)
        la_dist_data$start_time = hour(la_dist_data$start_time) + minute(la_dist_data$start_time)/60
        la_dist_data = data.table(arrange(la_dist_data, distance))
        la_dist_data = head(la_dist_data, 179013) # to account for outliers of distance


            #---------------------
            # PLOT for SF
            dist_sf = ggplot(data = Combined_lats_longs, aes(x=start_date, y=distance)) +
              geom_smooth(aes(colour = subscriber_type)) +
              labs(x = "Time of Day (24hr)", y = "Distance (meters)", title = "SF Trip Distance vs Day Time
(Customers vs. Subscribers)")


            dist_sf


            #----------------------
            #PLOT for LA
            dist_la = ggplot(data=la_dist_data, aes(x=start_time, y=distance)) +
              geom_smooth(aes(colour = passholder_type)) +
              labs(x = "Time of Day (24hr)", y = "Distance (meters)", title = "LA Trip Distance vs Day Time
(Passholder Types)")


            dist_la


# (pt 2) Frequency of Trips------------------------\
# create 3 subgroups within each of 5 subgroups dependent upon time and make boxplot of frequency
of trips
subgroups = as.data.table(arrange(Combined_lats_longs, start_date))
    sub_1_1 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >= 0) &
(start_date < 1.666667))"))
    sub_1_2 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >=
1.666667) & (start_date < 3.33333))"))
    sub_1_3 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >= 3.33333)
& (start_date < 5))"))
    sub_1 =
as.data.table(rbind(length(sub_1_1$start_date),length(sub_1_2$start_date),length(sub_1_3$start_date
)))


    sub_2_1 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >= 5) &
(start_date < 6.666667))"))
    sub_2_2 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >=
6.666667) & (start_date < 8.33333))"))
    sub_2_3 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >= 8.33333)
& (start_date < 10))"))
    sub_2 =
as.data.table(rbind(length(sub_2_1$start_date),length(sub_2_2$start_date),length(sub_2_3$start_date
)))


    sub_3_1 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >= 10) &
(start_date < 10.666667))"))
```

```
        sub_3_2 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >=
10.666667) & (start_date < 13.33333))"))

        sub_3_3 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >=
13.33333) & (start_date < 15))"))

        sub_3 =
as.data.table(rbind(length(sub_3_1$start_date),length(sub_3_2$start_date),length(sub_3_3$start_date
)))


        sub_4_1 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >= 15) &
(start_date < 16.666667))"))

        sub_4_2 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >=
16.666667) & (start_date < 18.33333))"))

        sub_4_3 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >=
18.33333) & (start_date < 20))"))

        sub_4 =
as.data.table(rbind(length(sub_4_1$start_date),length(sub_4_2$start_date),length(sub_4_3$start_date
)))


        sub_5_1 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >= 20) &
(start_date < 21.666667))"))

        sub_5_2 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >=
21.666667) & (start_date < 23.33333))"))

        sub_5_3 = as.data.table(sqldf("SELECT start_date FROM subgroups WHERE ((start_date >=
23.33333) & (start_date < 25))"))

        sub_5 =
as.data.table(rbind(length(sub_5_1$start_date),length(sub_5_2$start_date),length(sub_5_3$start_date
)))


freq_of_trips = as.data.table(rbind(sub_1,sub_2,sub_3,sub_4,sub_5))

temp = rep(letters)

temp = factor(temp)

temp = factor(temp[1:15])

levels(temp) = round((c(rep(seq(0,25, by = 1.66666666666666666666666666667)))),2)


all = as.data.table(cbind(freq_of_trips, temp))


freq_plot = ggplot(all, aes(x = temp, y = V1)) +

   geom_bar(stat = "identity") +

   labs(x = "Time of Day (24 hrs)", y = "Frequency of trips", title = "SF Frequency Plot of Trips Throughout
Average Day")

freq_plot


# could have wrote functions but just copy and pasted method for timing purposes
```

```
subgroups = as.data.table(arrange(la_dist_data, start_time))

sub_1_1 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 0) &
(start_time < 1.666667))"))

sub_1_2 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 1.666667) &
(start_time < 3.33333))"))

sub_1_3 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 3.33333) &
(start_time < 5))"))

sub_1 =
as.data.table(rbind(length(sub_1_1$start_time),length(sub_1_2$start_time),length(sub_1_3$start_time
)))


sub_2_1 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 5) &
(start_time < 6.666667))"))

sub_2_2 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 6.666667) &
(start_time < 8.33333))"))

sub_2_3 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 8.33333) &
(start_time < 10))"))

sub_2 =
as.data.table(rbind(length(sub_2_1$start_time),length(sub_2_2$start_time),length(sub_2_3$start_time
)))

sub_3_1 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 10) &
(start_time < 10.666667))"))

sub_3_2 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 10.666667) &
(start_time < 13.33333))"))

sub_3_3 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 13.33333) &
(start_time < 15))"))

sub_3 =
as.data.table(rbind(length(sub_3_1$start_time),length(sub_3_2$start_time),length(sub_3_3$start_time
)))


sub_4_1 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 15) &
(start_time < 16.666667))"))

sub_4_2 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 16.666667) &
(start_time < 18.33333))"))

sub_4_3 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 18.33333) &
(start_time < 20))"))

sub_4 =
as.data.table(rbind(length(sub_4_1$start_time),length(sub_4_2$start_time),length(sub_4_3$start_time
)))


sub_5_1 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 20) &
(start_time < 21.666667))"))

sub_5_2 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 21.666667) &
(start_time < 23.33333))"))

sub_5_3 = as.data.table(sqldf("SELECT start_time FROM subgroups WHERE ((start_time >= 23.33333) &
(start_time < 25))"))
```

```
sub_5 =
as.data.table(rbind(length(sub_5_1$start_time),length(sub_5_2$start_time),length(sub_5_3$start_time
)))


freq_of_trips = as.data.table(rbind(sub_1,sub_2,sub_3,sub_4,sub_5))
temp = rep(letters)
temp = factor(temp)
temp = factor(temp[1:15])
levels(temp) = round((c(rep(seq(0,25, by = 1.66666666666666666666666666667)))),2)


all2 = as.data.table(cbind(freq_of_trips, temp))


freq_plot2 = ggplot(all2, aes(x = temp, y = V1)) +
   geom_bar(stat = "identity") +
   labs(x = "Time of Day (24 hrs)", y = "Frequency of trips", title = "LA Frequency Plot of Trips Throughout
Average Day")
freq_plot2


#------------------------------------------------------------------------------------------------------------------------------------
----
# 6.)
       # Refer to previously created data frame with all necessary items for calculation


       bearings = as.data.table(bearing(p1, p2))
       names(bearings) = "bearings"
       Combined_lats_longs$bearings = bearings




       bearings_sf = ggplot(data = Combined_lats_longs, aes(x=start_date, y=bearings)) +
          geom_smooth(aes(colour = subscriber_type)) +
          labs(x = "Time of Day (24hr)", y = "Bearing Angle (Degrees)", title = "SF Bearing Angle vs Day Time
(Customers vs. Subscribers)")


       bearings_sf
```