# Ford GoBike Data Analysis

CODE

This analysis will provide some insights from the Ford GoBike data in San Francisco area.

## Packages Required

HIDE

```
library(readr)
library(tidyverse)
library(dplyr)
library(magrittr)
library(lubridate)
library(chron)
library(scales)
library(ggmap)
library(ggrepel)
library(xts)
library(forecast)
library(tseries)
library(bannerCommenter)
```

- `readr` : to provide a fast way to read .csv files
- `tidyverse` : to clean, reorganize and visualize datasets
- `dplyr` : for data manipulation
- `magrittr` : to provide mechanism for commands with pipe operator
- `lubridate` : to manipulate date
- `chron` : to create chronological objects
- `scales` : plot scaling method
- `ggmap` : for creating maps
- `ggrepel` : to prevent overlapped labels
- `xts` : time series analysis
- `tseries` : time series analysis
- `forecast` : for forecasting
- `rmarkdown` : for creating better rmd
- `knitr` : for dynamic report generation
- `bannercommenter` : to create comment area

# Data Preparation

## Original datasets

The data set about Ford GoBike trips was accessed via It's official site (https://www.fordgobike.com/) There are 5 data files:

- bike share in 2017 since June 28
- bike share in 2018 January, Feburary, March and April

## Read data and some cleaning

Final dataset: total (2017 & 2018 combined)

HIDE

```
# import data
setwd("E:/GoogleExpress")
Jan2018 <- read_csv("201801-fordgobike-tripdata.csv")
Feb2018 <- read_csv("201802-fordgobike-tripdata.csv")
Mar2018 <- read_csv("201803-fordgobike-tripdata.csv")
Apr2018 <- read_csv("201804-fordgobike-tripdata.csv")
x2017 <- read_csv("2017-fordgobike-tripdata.csv")
# observation: 2017 data starts from June 28


# combine 2018 month 1-4 to get x2018
rbind(Jan2018, Feb2018, Mar2018, Apr2018) %>%
  arrange(start_time) -> x2018
# combine 2017 and 2018 data
rbind(x2017, x2018[ ,1:15]) %>%
  arrange(start_time) -> total
```

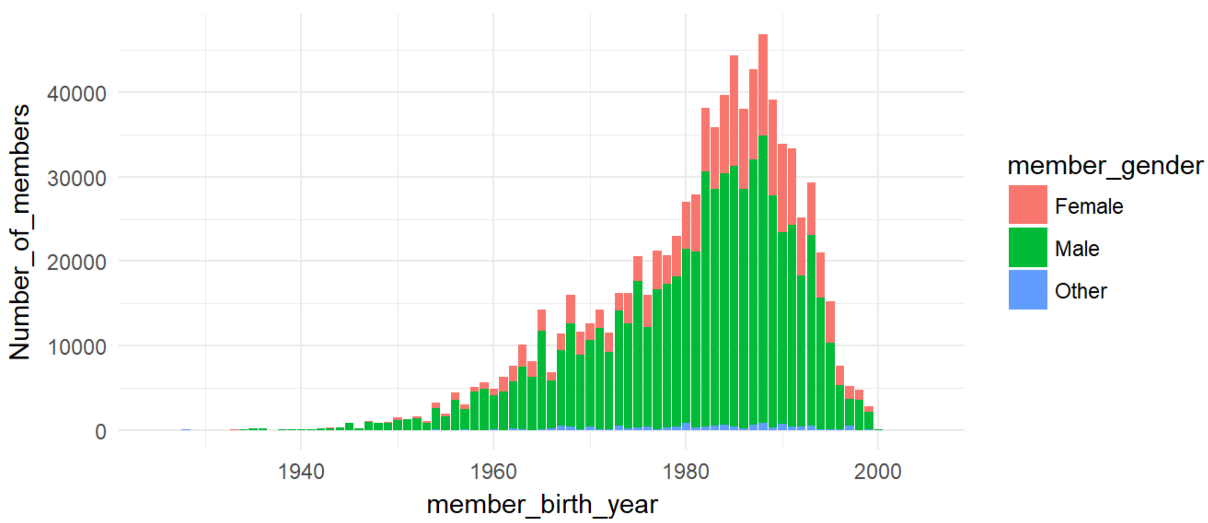
# Data Analysis

## Member demographics

### Age & Gender

HIDE

```
total %>%
  select(member_birth_year, member_gender) %>%
  group_by(member_birth_year, member_gender) %>%
  summarise(Number_of_members = n()) %>%
  arrange(desc(member_birth_year)) %>%
  ggplot(aes(x=member_birth_year, y=Number_of_members, fill=member_gender)) +
  geom_bar(stat = "identity") +
  scale_x_continuous(limits = c(1925, 2005)) +
  theme_minimal()
```



Majority of the users of GoBike fall in the 25 - 38 age bracket. GoBike's relatively young customer base reflects that biking is more preferred among the new generations who embrace sharing economy and maybe at the early stages of their career (lower income level).

Number of male users is around 3 times of female users, across all age ranges. This is in consistency with gender ratio in the area.

# Trip Duration

## Duration distribution in 2017

HIDE

```
summary17 <- summary(x2017$duration_sec)
summary17
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##      61     382     596    1099     938   86369
```

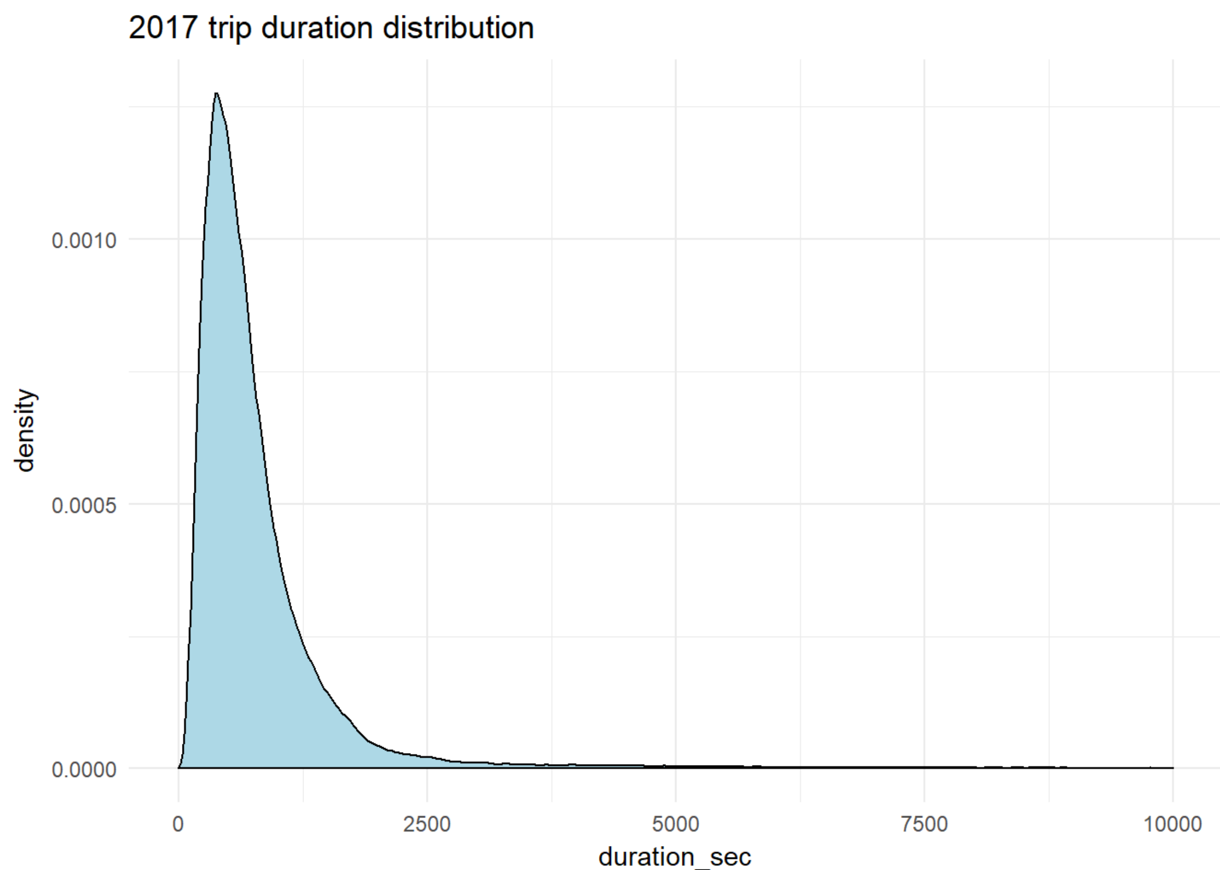
HIDE

```
sd17 <- sd(x2017$duration_sec)
print(paste("standard deviation:", sd17, sep=" "))
```

```
## [1] "standard deviation: 3444.14645124744"
```

HIDE

```
options(scipen=999)
# 2017 density plot
den_17 <- ggplot(x2017, aes(x=duration_sec)) +
  geom_density(fill="lightblue") +
  scale_x_continuous(limits = c(0,10000)) +
  ggtitle("2017 trip duration distribution") +
  theme_minimal()
den_17
```




### Duration distribution in 2018

HIDE

```
# 2018
summary18 <- summary(x2018$duration_sec)
summary18
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    61.0   353.0   552.0   877.1   858.0 86366.0
```

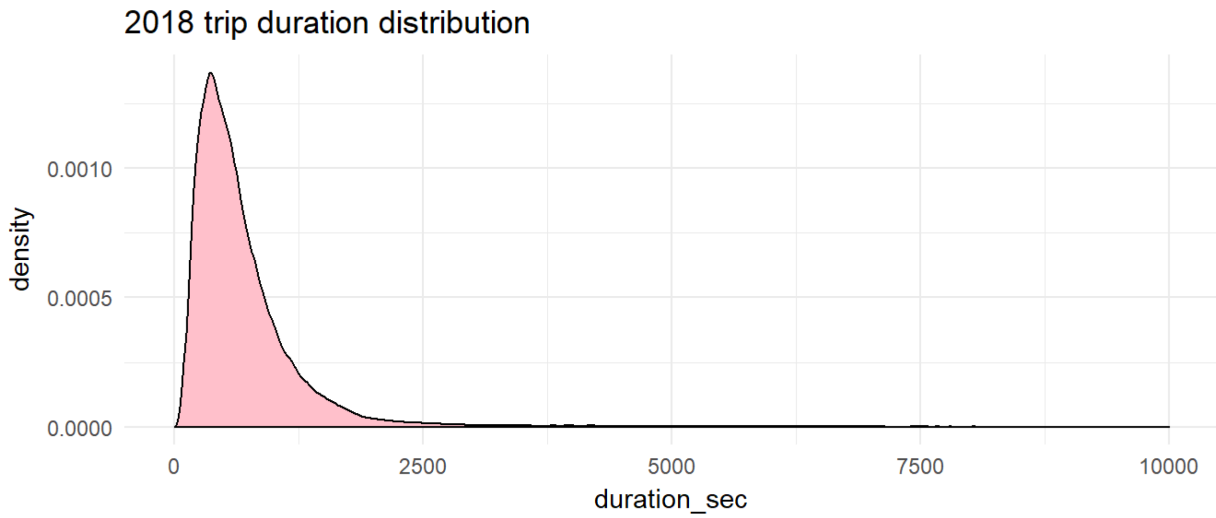
HIDE

```
sd18 <- sd(x2018$duration_sec)
print(paste("standard deviation:", sd18, sep=" "))
```

```
## [1] "standard deviation: 2616.34573170494"
```

HIDE

```
# 2018 density plot
den_18 <- ggplot(x2018, aes(x=duration_sec)) +
  geom_density(fill="pink") +
  scale_x_continuous(limits = c(0,10000)) +
  ggtitle("2018 trip duration distribution") +
  theme_minimal()
den_18
```

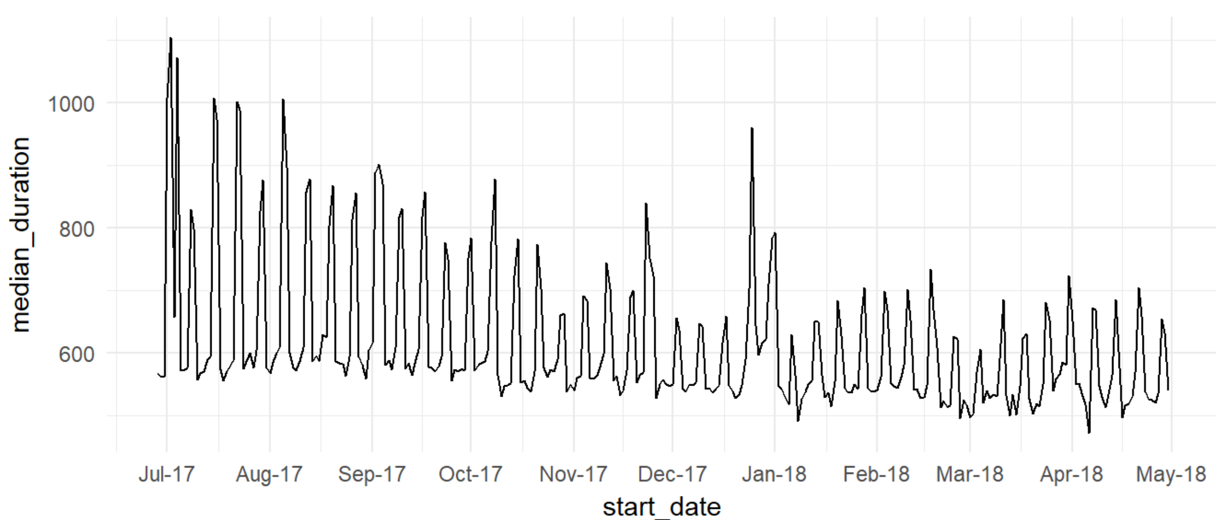



As shown from the density plots, most trips last for less than 2500 seconds, which is approximately 42 minutes. The majority of bike users spent around 400 seconds for their trip, which translates to 6.7 minutes. Compared with 2017, in 2018 people's trip duration decreased significantly, with the mean from 1099 seconds to 877 seconds. The variation also became smaller, with standard deviation dropping from 3444 to 2616. People are spending shorter amount of time per trip, which can be a result of more bike stations.

### Median duration by day

HIDE

```
total$start_date <- as.Date(total$start_time)
total %>%
  group_by(start_date) %>%
  summarise(median_duration = median(duration_sec)) -> series1
# duration - time series
p1 <- ggplot(series1, aes(x=start_date, y=median_duration)) +
  geom_line() +
  scale_x_date(breaks = date_breaks("month"), labels = date_format("%b-%y")) +
  theme_minimal()

p1
```




As we can observe from the line chart, there's a strong seasonal trend in the median riding duration. This can be explained by that people tend to enjoy biking more during weekends.

# Busiest dates & times

## Busiest dates

Busiest dates are determined by number of rides initiated on the date

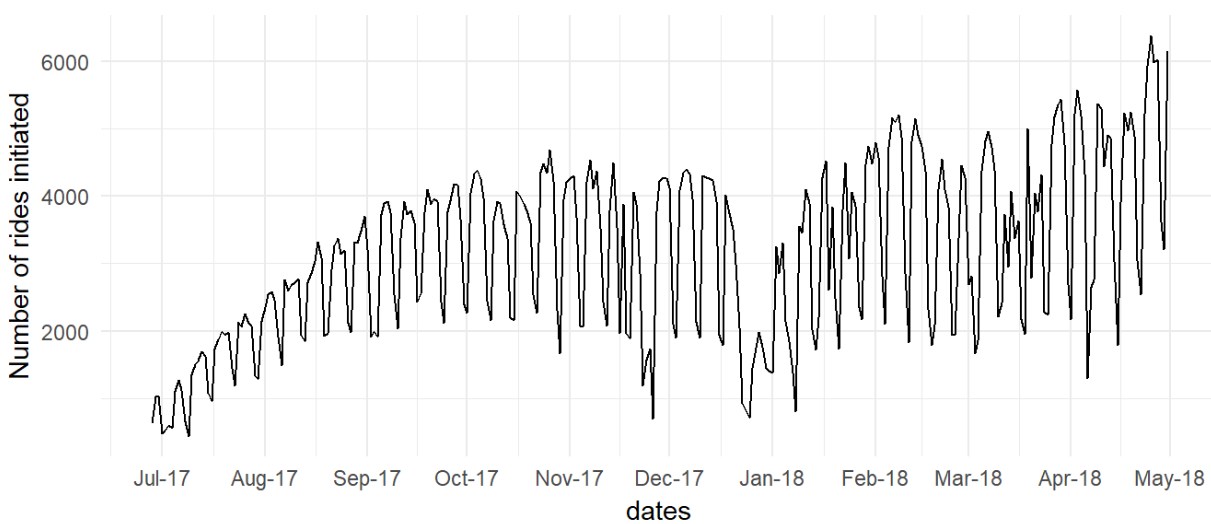
HIDE

```
dates <- total$start_date
frequencies <- as.data.frame(table(dates))
frequencies$dates <- as.Date(frequencies$dates)

busiest <-top_n(frequencies, 10, Freq)
busiest
```

```
##         dates Freq
## 1  2018-03-28 5356
## 2  2018-03-29 5432
## 3  2018-04-03 5566
## 4  2018-04-09 5365
## 5  2018-04-10 5296
## 6  2018-04-24 5927
## 7  2018-04-25 6377
## 8  2018-04-26 5978
## 9  2018-04-27 6020
## 10 2018-04-30 6140
```

HIDE

```
# time series plot of number of rides
frequencies %>%
  ggplot(aes(x=dates, y=Freq)) +
  geom_line() +
  scale_x_date(breaks = date_breaks("month"), labels = date_format("%b-%y")) +
  ylab("Number of rides initiated") +
  theme_minimal() -> p2
p2
```

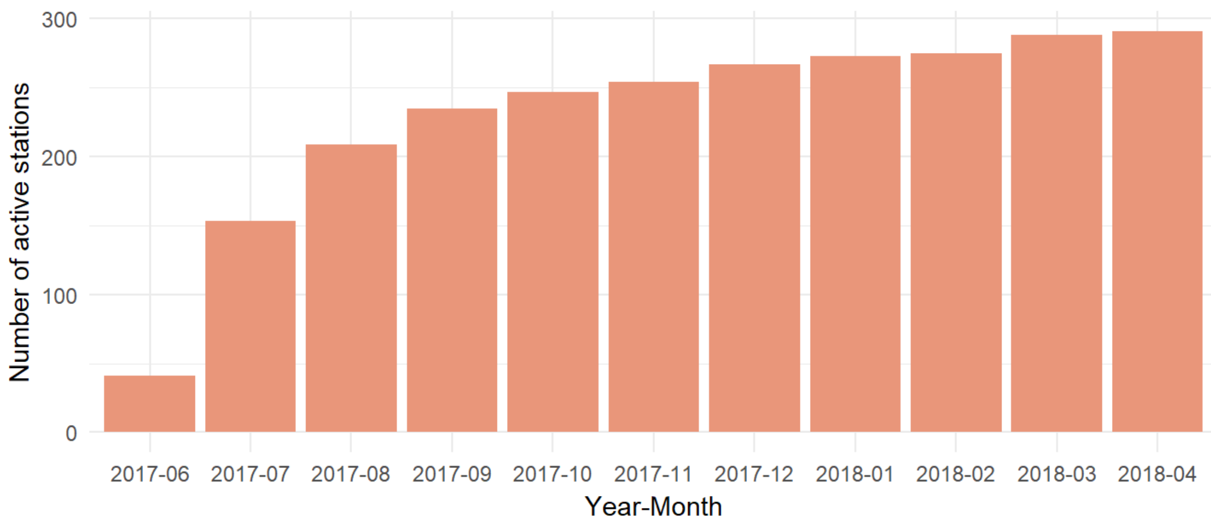



The busiest dates are all recent days, and from the line chart we can observe a general growing trend. We can try to explain this by seeing the number of stations opened in the time period. The assumed situation is that more stations were opened in the SF ares as time went.

## Number of active stations

HIDE

```
total %>%
  mutate(m=floor_date(start_date, "month")) %>%
  group_by(m) %>%
  summarise(n_distinct((start_station_id))) -> series2

series2$m <- format(as.Date(series2$m, format="%Y/%m/%d"),"%Y-%m")
colnames(series2)[2] <- "nstation"


ggplot(series2, aes(x=m, y=nstation)) +
  geom_bar(stat = 'identity', fill="darksalmon") +
  xlab("Year-Month") + ylab("Number of active stations") +
  theme_minimal() -> p3
p3
```



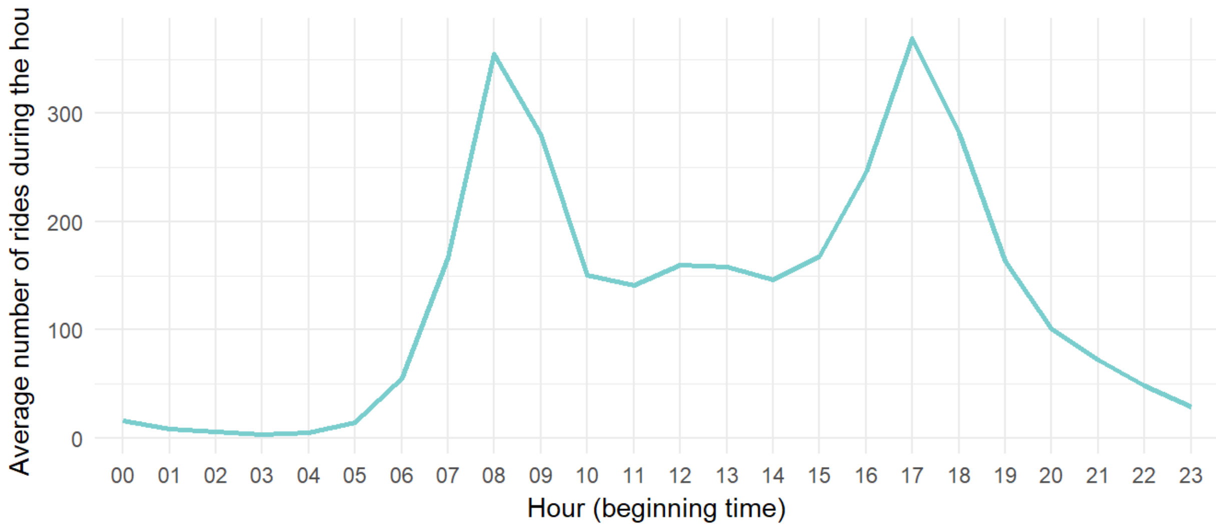

As can be seen from the bar chart, number of active stations increased over time. This is consistent with the growing rides. The fact that more stations were opened is attibutable to the increased trips.

## Busiest times

Busiest times are determined by average hourly number of rides initiated, across the whole period (2017.6 - 2018.4)

HIDE

```
# create hour variable
total$hour <- format(total$start_time,format="%H")
# calculate average number of rides in each hour
total %>%
  group_by(hour, start_date) %>%
  tally() -> hours
hourly_rides <- aggregate(hours[ ,3], list(hours$hour), mean)

p3 <- ggplot(hourly_rides, aes(x=Group.1, y=n, group=1)) +
  geom_line(color = "darkslategray3", size = 1) +
  xlab("Hour (beginning time)") + ylab("Average number of rides during the hour") +
  theme_minimal()
p3
```



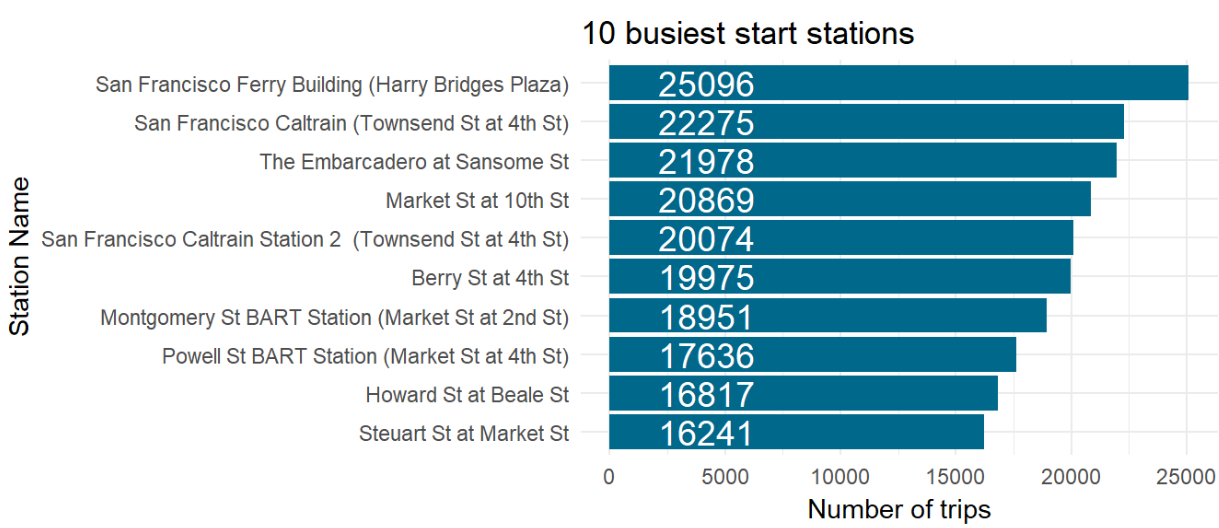

8:00-9:00, and 17:00-18:00 are 2 peaks of bike use. We can deduce that many people use shared bikes to commute to work.

# Most common stations

## Top 10 starting stations

HIDE

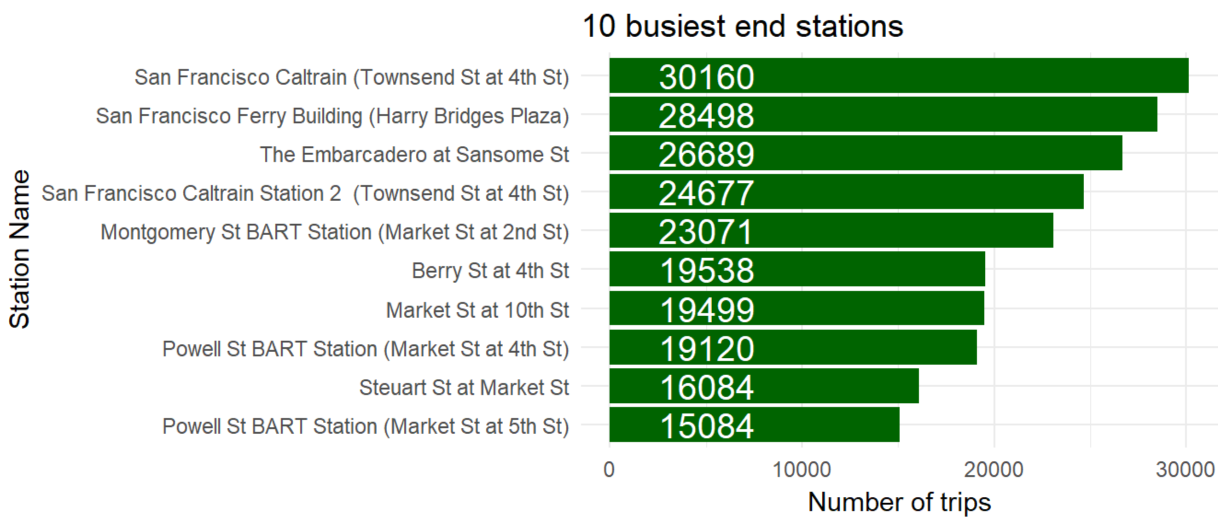
```
total %>%
  group_by(start_station_name) %>%
  summarise(number_of_rides = n()) %>%
  arrange(desc(number_of_rides)) %>%
  head(10) -> top_10_start

ggplot(data = top_10_start, aes(x=reorder(start_station_name,number_of_rides),y=number_of_rides)) +
  geom_bar(stat = "identity", fill="deepskyblue4") +
  geom_text(aes(x = reorder(start_station_name,number_of_rides), y = 1,
              label = paste(number_of_rides)),
            hjust=-0.5, vjust=.5, size = 5, colour = 'white') +
  labs(x="Station Name", y="Number of trips", title="10 busiest start stations") +
  coord_flip() +
  theme_minimal() -> p4
p4
```




## Top 10 ending stations

HIDE

```
total %>%
  group_by(end_station_name) %>%
  summarise(number_of_rides = n()) %>%
  arrange(desc(number_of_rides)) %>%
  head(10) -> top_10_end

ggplot(data = top_10_end, aes(x=reorder(end_station_name,number_of_rides),y=number_of_rides)) +
  geom_bar(stat = "identity", fill="darkgreen") +
  geom_text(aes(x = reorder(end_station_name,number_of_rides), y = 1,
              label = paste(number_of_rides)),
          hjust=-0.5, vjust=.5, size = 5, colour = 'white') +
  labs(x="Station Name", y="Number of trips", title="10 busiest end stations") +
  coord_flip() +
  theme_minimal() -> p5
p5
```




## Trip map

Knowing the Top 10 common start and end stations, we can further look into the common routes. There are 3 major areas involved in the data - San Francisco, Oakland and San Jose. Here we only map the routes in SF area.

Top popular stations are ranked by the sum of starting trips and ending trips.

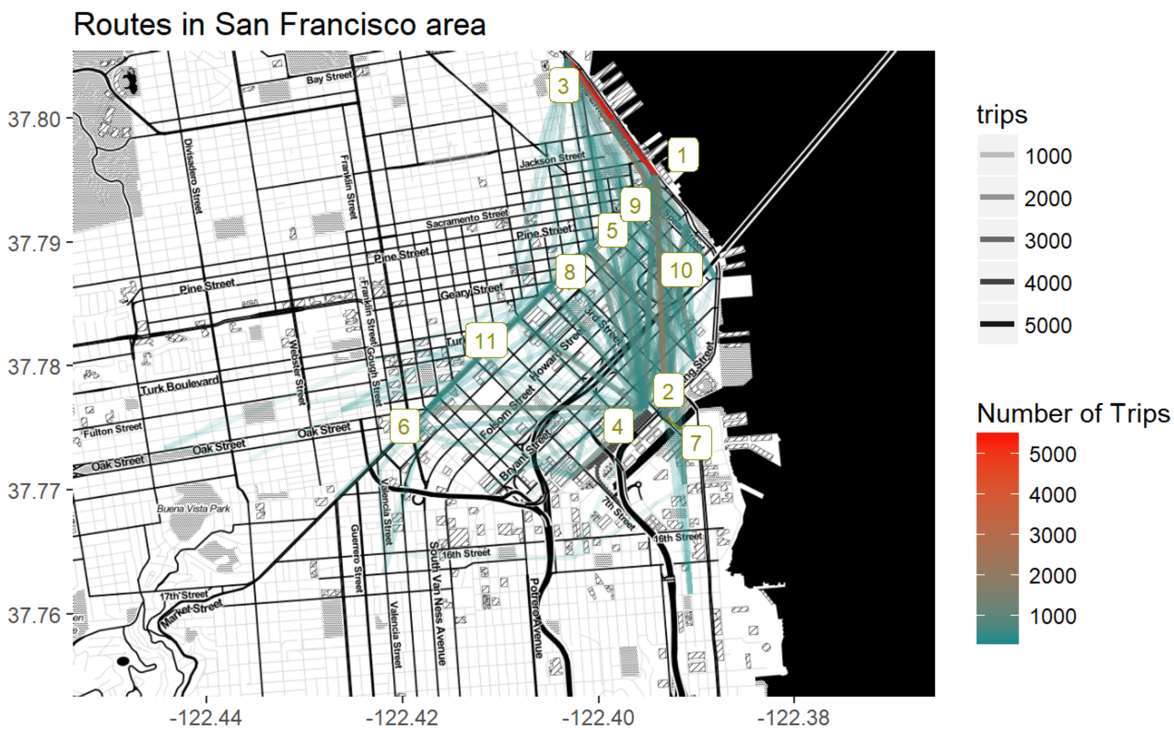
HIDE

```r
# trips & station list
total%>%
  select(start_station_id,end_station_id) %>%
  group_by(start_station_id, end_station_id) %>%
  summarise(trips=n()) %>%
  left_join(total[4:11],by = c("start_station_id", "end_station_id")) %>%
  filter(trips >= 300) %>%
  unique() -> trips


# get map base
SF <- c(-122.445,37.770,-122.375,37.805)
Map <- get_map(location=SF,
                 source="stamen", maptype="toner", crop=FALSE)
SFmap <- ggmap(Map)
###################################################################
##                Routes map (focusing on SF area)                ##
###################################################################
# label busiest stations in the map
# get busiest list
colnames(top_10_start)[1] <- "station_name"
colnames(top_10_end)[1] <- "station_name"
rbind(top_10_start, top_10_end) %>%
  unique() %>%
  group_by(station_name) %>%
  mutate(total_trips = sum(number_of_rides)) %>%
  select(station_name, total_trips) %>%
  unique() %>%
  arrange(desc(total_trips)) -> top_list

top_list$rank <- seq.int(nrow(top_list))
colnames(top_list)[1] <- "start_station_name"

# find lat & long for top list
merge(x=top_list, y=trips, by="start_station_name", all.x=T) %>%
  select(rank, start_station_name, total_trips, start_station_latitude, start_station_longitude) %>%
  unique() %>%
  arrange(desc(total_trips)) %>%
  `colnames<-`(c("rank", "station_name", "total_trips",
                 "station_latitude", "station_longitude")) -> top_list

print(top_list[1:3])
```

```
##    rank                                         station_name
## 1     1        San Francisco Ferry Building (Harry Bridges Plaza)
## 2     2            San Francisco Caltrain (Townsend St at 4th St)
## 3     3                             The Embarcadero at Sansome St
## 4     4 San Francisco Caltrain Station 2  (Townsend St at 4th St)
## 5     5          Montgomery St BART Station (Market St at 2nd St)
## 6     6                                    Market St at 10th St
## 7     7                                      Berry St at 4th St
## 8     8              Powell St BART Station (Market St at 4th St)
## 9     9                                 Steuart St at Market St
## 10   10                                   Howard St at Beale St
## 11   11              Powell St BART Station (Market St at 5th St)
##    total_trips
## 1        53594
## 2        52435
## 3        48667
## 4        44751
## 5        42022
## 6        40368
## 7        39513
## 8        36756
## 9        32325
## 10       16817
## 11       15084
```

HIDE

```
p6 <- SFmap +
  geom_segment(data=trips, aes(x=start_station_longitude, xend=end_station_longitude,
                               y=start_station_latitude, yend=end_station_latitude,
                               alpha=trips,color=trips), size=1.2) +
  scale_size_continuous(range = c(1,12)) +
  scale_colour_gradientn(colors=c("darkcyan", "red"),
                         limits=c(300, max(trips$trips)), name="Number of Trips") +
  geom_label_repel(data=top_list,aes(x=station_longitude, y=station_latitude,label=rank),
                   color="yellow4", size=3) +
  labs(x="", y="", title="Routes in San Francisco area")

p6
```



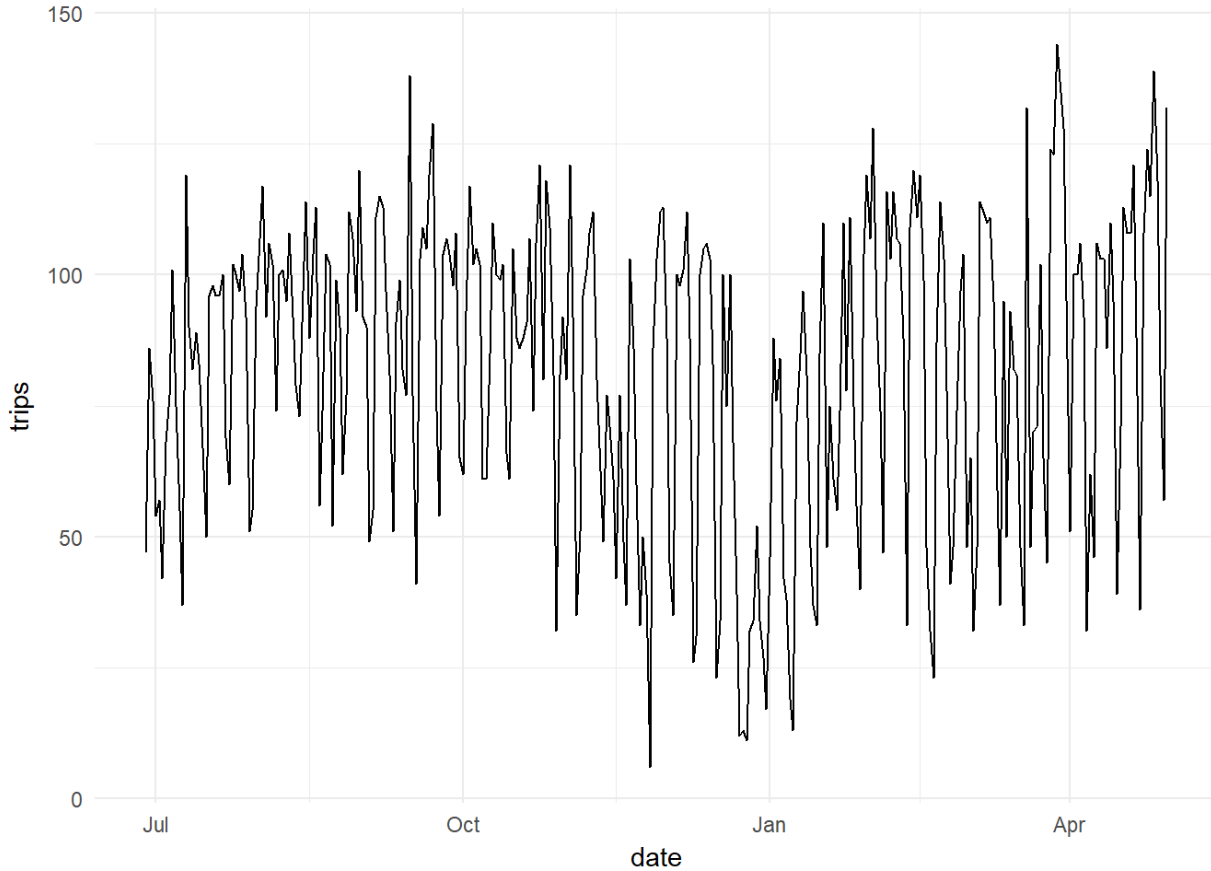

## Forecast for mot popular station

San Francisco Ferry Building (Harry Bridges Plaza) is the most popular station, with 53594 trips started and ended in the data period.

In the final part, we look at a 2-week forecast of number of trips initiated from this station.

HIDE

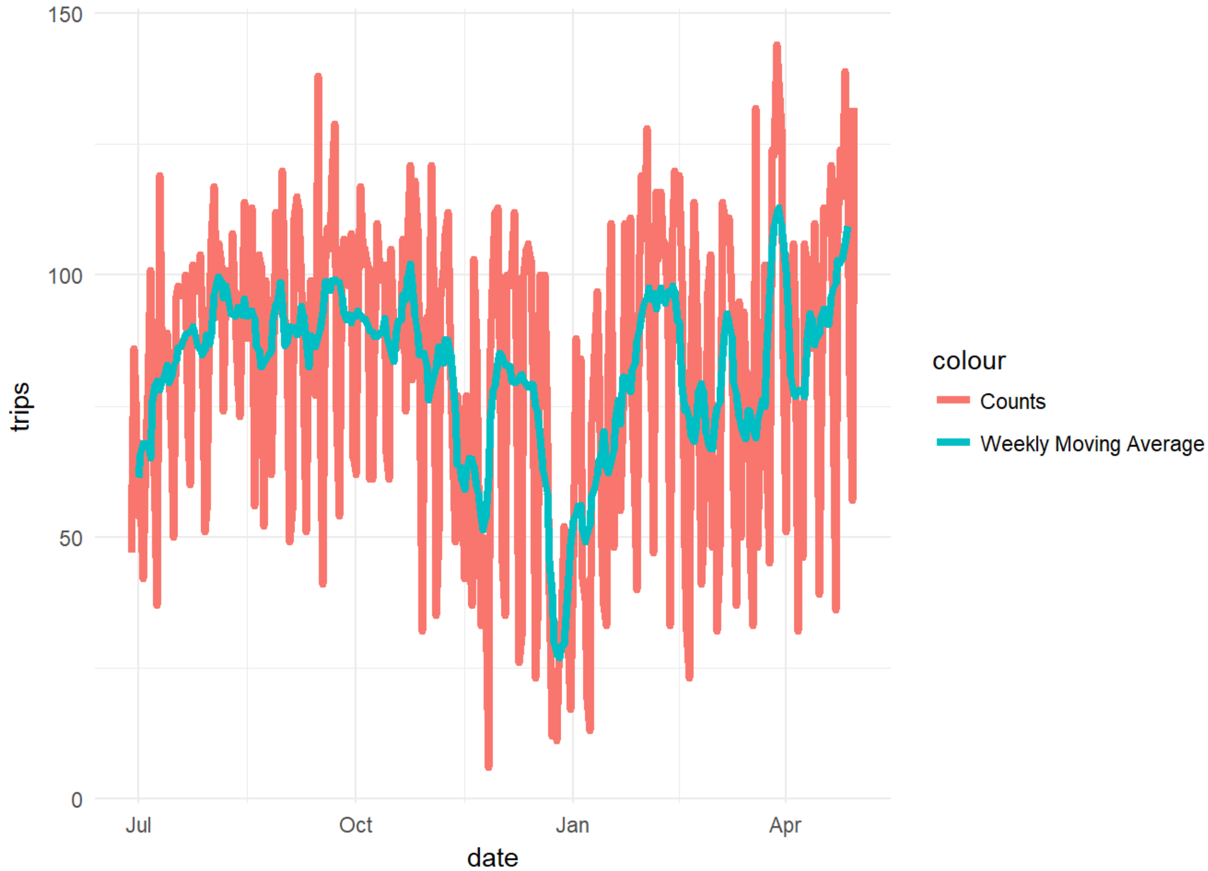
```
total %>%
  filter(start_station_name=="San Francisco Ferry Building (Harry Bridges Plaza)") %>%
  select(start_date)  -> ts
as.data.frame(table(ts)) %>%
  `colnames<-`(c("date", "trips")) -> series3

series3$date <- as.Date(series3$date, format="%Y-%m-%d")
# create time series
ts <- ts(series3[, c('trips')])

p7 <- ggplot(series3, aes(x=date, y=trips, group=1))+
  geom_line() +
  theme_minimal()
p7
```




From the series plot, there's no obvious outliers, so we use the original data to forecast weekly moving average.

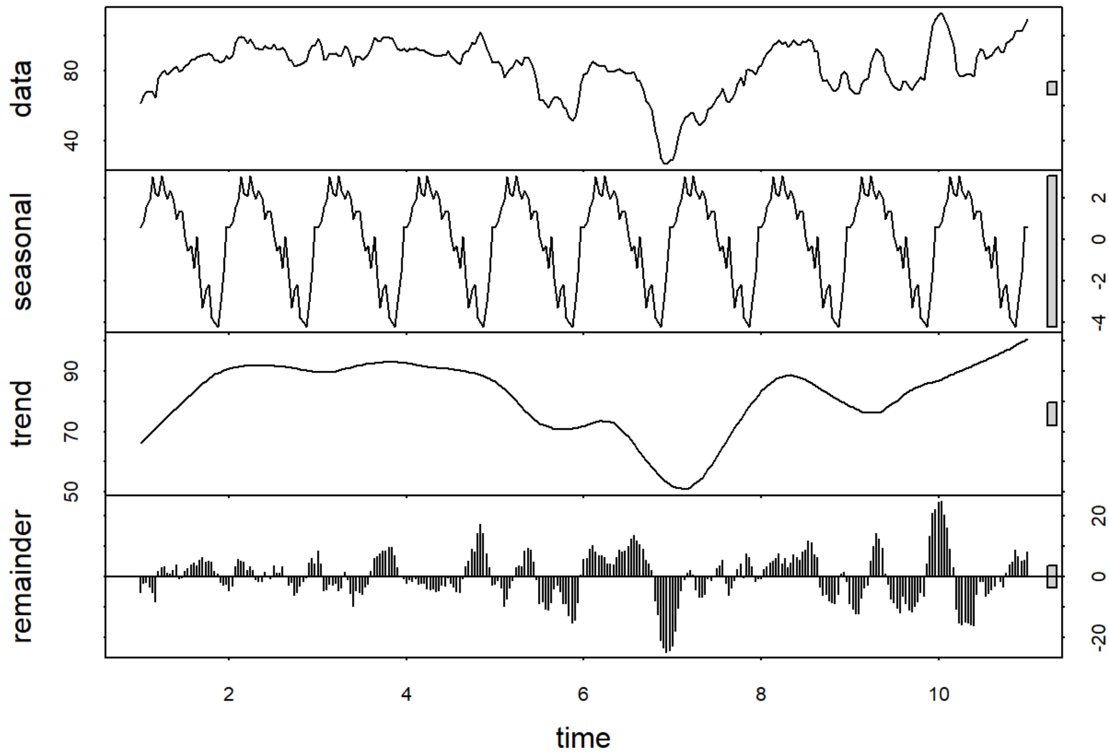
HIDE

```
series3$trips_ma = ma(series3$trips, order=7)
ggplot(size = 2) +
  geom_line(data = series3, aes(x = date, y = trips, color = "Counts"), size = 1.5) +
  geom_line(data = series3, aes(x = date, y = trips_ma,   color = "Weekly Moving Average"), size = 1.5) +
  theme_minimal() -> p8
p8
```




HIDE

```
# calculate seasonal component of series using monthly period
series_m = ts(na.omit(series3$trips_ma), frequency=30)
decomp = stl(series_m, s.window="periodic")
deseasonal_trips <- seasadj(decomp)
plot(decomp)
```




Fitting ARIMA model & forecast

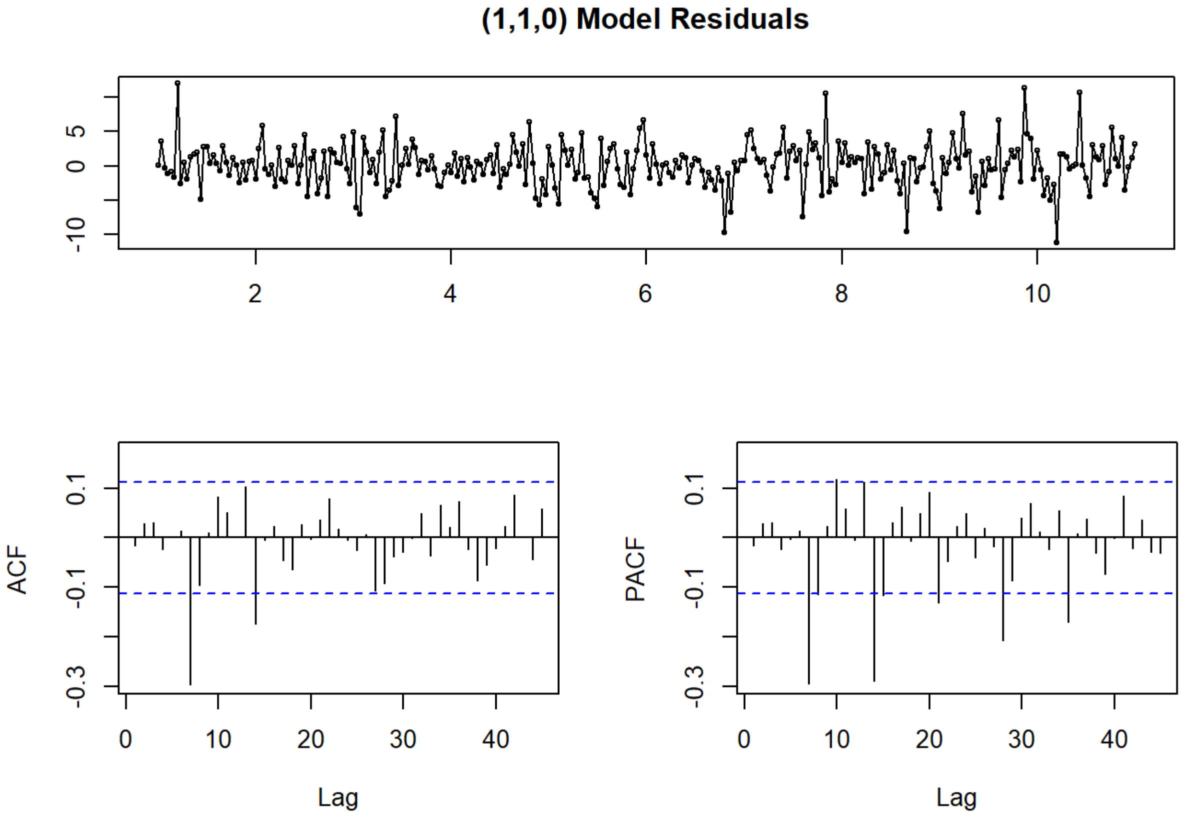
HIDE

```
# auto ARIMA
auto.arima(deseasonal_trips, seasonal=FALSE)
```

```
## Series: deseasonal_trips
## ARIMA(1,1,0)
##
## Coefficients:
##          ar1
##       0.4304
## s.e.  0.0522
##
## sigma^2 estimated as 10.31:  log likelihood=-775.21
## AIC=1554.41   AICc=1554.45   BIC=1561.82
```

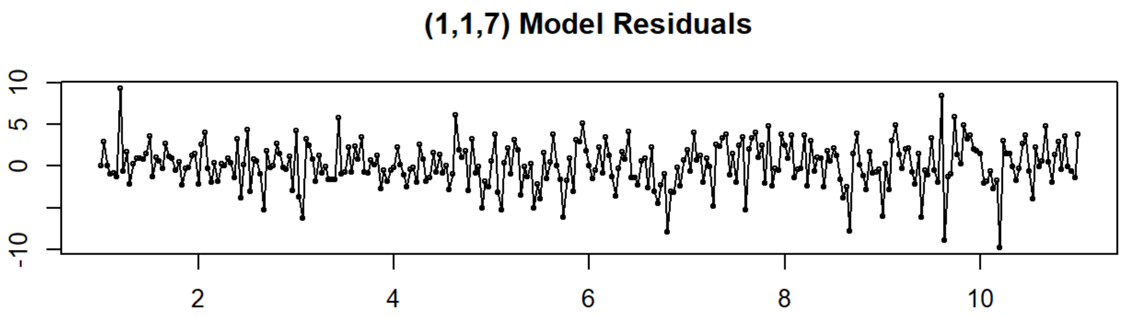
HIDE

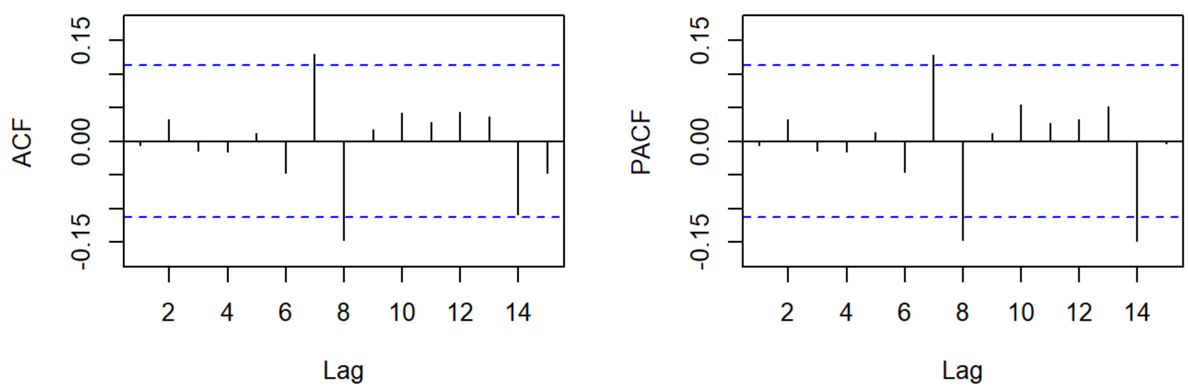
```
# fit the model
fit<-auto.arima(deseasonal_trips, seasonal=FALSE)
tsdisplay(residuals(fit), lag.max=45, main='(1,1,0) Model Residuals')
```




HIDE

```
# ACF and PACF still have spikes at lag 7, so we can try MA(7)
fit2 = arima(deseasonal_trips, order=c(1,1,7))

tsdisplay(residuals(fit2), lag.max=15, main='(1,1,7) Model Residuals')
```




HIDE

```
# the residuals are close to white noise with ARIMA(1,1,7)

arima(x = deseasonal_trips, order = c(1, 1, 7))
```

```
##
## Call:
## arima(x = deseasonal_trips, order = c(1, 1, 7))
##
## Coefficients:
##          ar1     ma1     ma2     ma3     ma4     ma5     ma6      ma7
##       0.2894  0.0863  0.1317  0.1644  0.1020  0.0976  0.1375  -0.7696
## s.e.  0.0954  0.0767  0.0521  0.0524  0.0554  0.0598  0.0522   0.0499
##
## sigma^2 estimated as 7.066:  log likelihood = -723.78,  aic = 1465.55
```

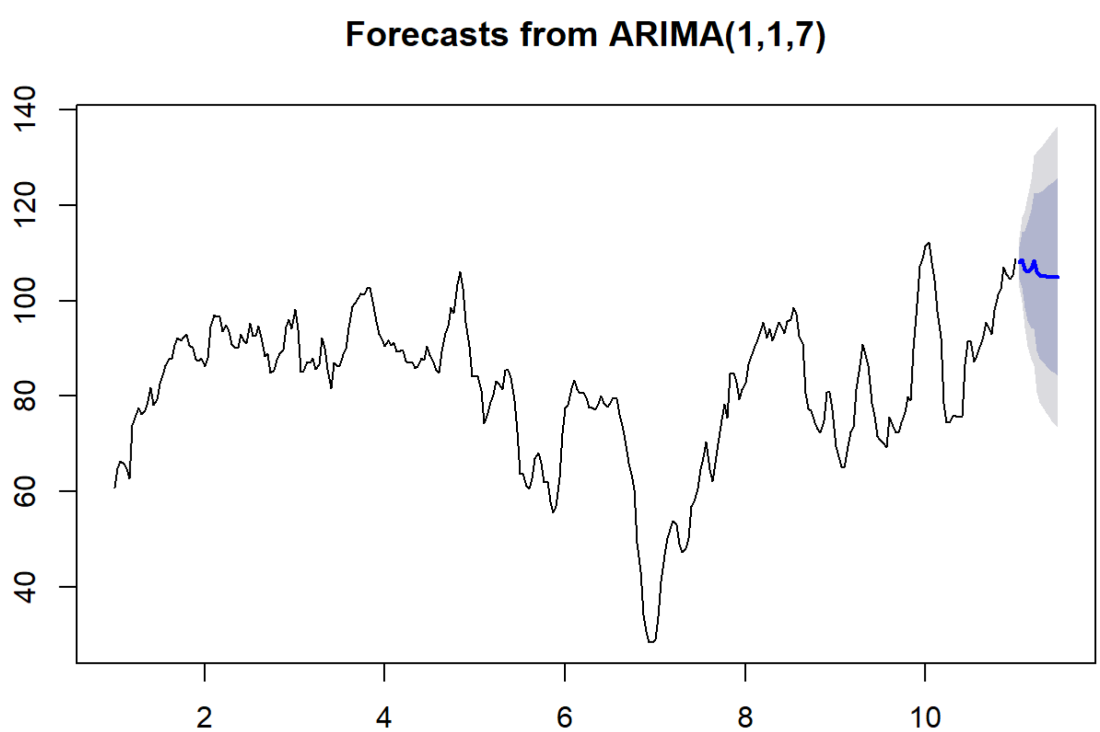
HIDE

```
# forecast
fcast <- forecast(fit2, h=14)
print(fcast)
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 11.03333       107.9440 104.53738 111.3505 102.73405 113.1539
## 11.06667       108.6738 102.87989 114.4678  99.81276 117.5349
## 11.10000       106.5139  98.52124 114.5066  94.29016 118.7377
## 11.13333       106.0970  95.91824 116.2758  90.52992 121.6641
## 11.16667       106.6304  94.34412 118.9166  87.84017 125.4206
## 11.20000       108.4118  94.07931 122.7444  86.49213 130.3316
## 11.23333       105.9966  89.58584 122.4074  80.89849 131.0947
## 11.26667       105.2976  87.97408 122.6211  78.80357 131.7916
## 11.30000       105.0953  87.10796 123.0826  77.58607 132.6045
## 11.33333       105.0367  86.46142 123.6120  76.62826 133.4451
## 11.36667       105.0198  85.88880 124.1507  75.76149 134.2780
## 11.40000       105.0148  85.34790 124.6818  74.93685 135.0928
## 11.43333       105.0134  84.82583 125.2010  74.13917 135.8877
## 11.46667       105.0130  84.31818 125.7079  73.36300 136.6630
```

HIDE

```
plot(fcast)
```




Above are forecasted number of trips in the following 2 weeks after the data. This is a relatively naive model, and the results can be validated when the data becomes available.