DATA 698 Final Research Project

Philip Tanofsky

2022-11-13

Contents

Abstract

Keywords: xx, xx, xx,

P1: - Contents - Abstract - Keywords

Introduction

The Author

Philip Tanofsky is a software engineer lead and a Master's of Data Science candidate at City University of New York - School of Professional Studies. Philip lives with his roommate Speck, a terrier mix, in Brooklyn, New York. Philip would like to thank Dr. Bailo and Alex XXX with support and guidance throughout the process and Evi Wiley for a lifetime of support.

The Problem

Pull from my problem statement to complete this section 1. Determine the flow of bicycles throughout the Citi Bike system in New York City to identify neighborhoods with a surplus of bicycles or a dearth of bicycle availability. 2. What are the factors that drive the usage of bicycles and in turn impact the level of availability at a given station or cluster. 3. Determine the best model to accurately model the availability of bicycles based on day of the week, time of day, location, and number of bicycles needed?

Literature Review

- Lit Review start P3-P4: Lit Review Already submitted

Methodology

P5: - Methodology Attempt to answer the following questions?

Data Collection & Preprocessing

- Data Collection and Preprocessing P6-P8: Collection and Preprocessing
- Likely pretty light for me as this decent data from Citi Bike

Data Collection

Citi Bike provides individual bike trip data on a monthly basis available at https://ride.citibikenyc.com/system-data. The September 2022 bike trip data for New York City was downloaded and unzipped. The dataset contains 13 variables for each bike trip originating at a NYC-based location. A note on the system data page indicates trips taken by staff to service or inspect the system have been removed from the dataset. Also, any trips below 60 seconds have also been omitted. With this preprocessing by the data maintainers, the remaining trips are considered to be valid bike trips.

- **ride.id:** Unique identifier of the bike trip
- **rideable.type:** Factor variable - classic, electric, and docked
- **started.at:** Timestamp of trip departure
- **ended.at:** Timestamp of trip arrival
- **start.station.name:** Name of departure docking station
- **start.station.id:** Unique identifier of departure docking station
- **end.station.name:** Name of arrival docking station
- **end.station.id:** Unique identifier of arrival docking station
- **start.lat:** Latitude of departure location
- **start.lng:** Longitude of departure location
- **end.lat:** Latitude of arrival location
- **end.lng:** Longitude of arrival location
- **member.casual:** Factor variable for user type - member or casual

Based on the `started.at` and `ended.at` variables, four variables are derived for each bike trip.

- **day:** Day of the month
- **start.hour:** Hour of the trip departure
- **weekday:** Day of the week for the trip
- **trip.duration:** Duration of bike trip in minutes.

In order to track the bike availability at each docking station, an API call is made to the Citi Bike General Bikeshare Feed (https://github.com/MobilityData/gbfs/blob/master/gbfs.md) for the Station Status JSON every 15 minutes from Monday, October 31 through Sunday, November 14 using a chron script.

Read in the data and check if legacy and station.id are ever different and then derived column for

- **legacy.id:**: Legacy unique identifier of the docking station
- **station.status:** Factor variable - active or out of service
- **num.bikes.available:** Number of classic bikes available

- **num.ebikes.available:** Number of eBikes available
- **station.id:** Unique identifier of the docking station
- **num.total.bikes.available:** Sum of 'num.bikes.available' and 'num.ebikes.available'

The NYC Open Data (free public data published by New York City agencies and partners) provides a GeoJSON file for the polygons defining each neighbhorhood in NYC according for the 2010 Neighborhood Tabulation Areas (NTAs). Each NTA is associated with one of the five NYC boroughs. (https://data.cityofnewyork.us/City-Government/2010-Neighborhood-Tabulation-Areas-NTAs-/cpf4-rkhq)

The elevation of each Citi Bike docking station is determined using the R library `elevatr` based on the latitude and longitude of each station. The elevation is defined in meters above sea level.

- **elevation: Units above sea level**
- **elev.units: Unit of measurement for elevation**

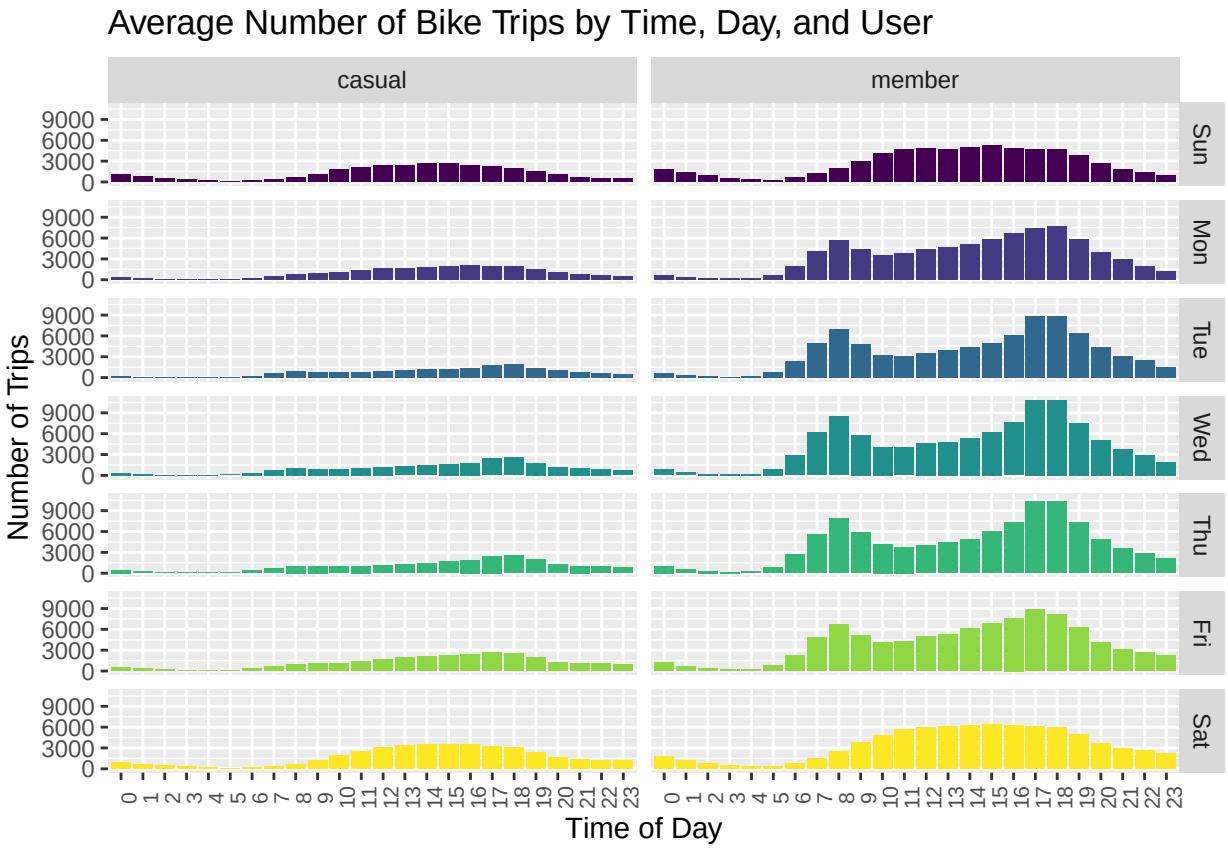
Data Preprocessing

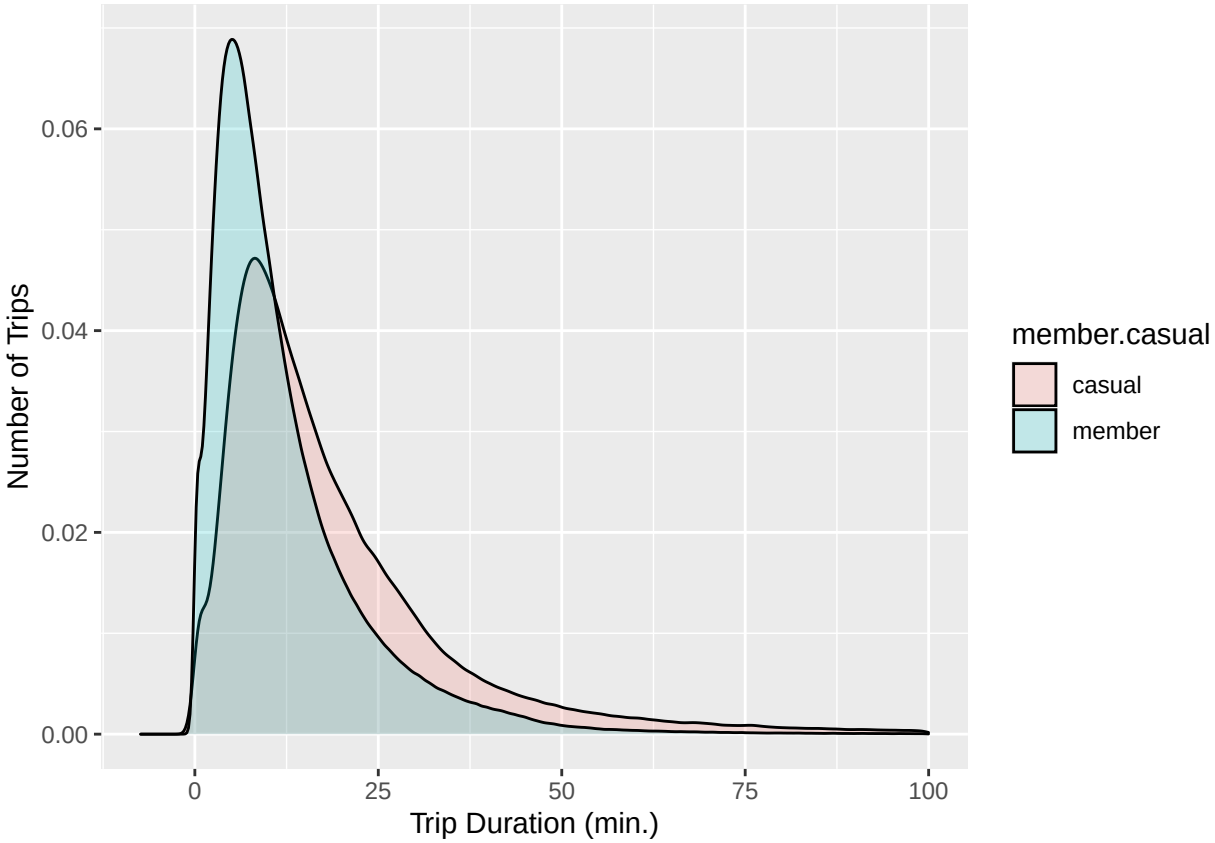
Upon initial inspection of the 3,507,123 bike trips in September 2022, a total of 8,012 entries do not have an `end.station.id` and `end.station.name` listed. Of that count, 3,838 do not contain an `end.lat` and `end.lng` values. These 8,012 without a defined destination docking station will be defined as 'Abandoned' meaning the rider did not proper dock the bike. For this purpose, the `end.lat` and `end.lng` will be removed as the abandoned bikes temporarily remove a bike from the bikeshare system. Another rider cannot rent an abandoned bike until the bike is properly docked.

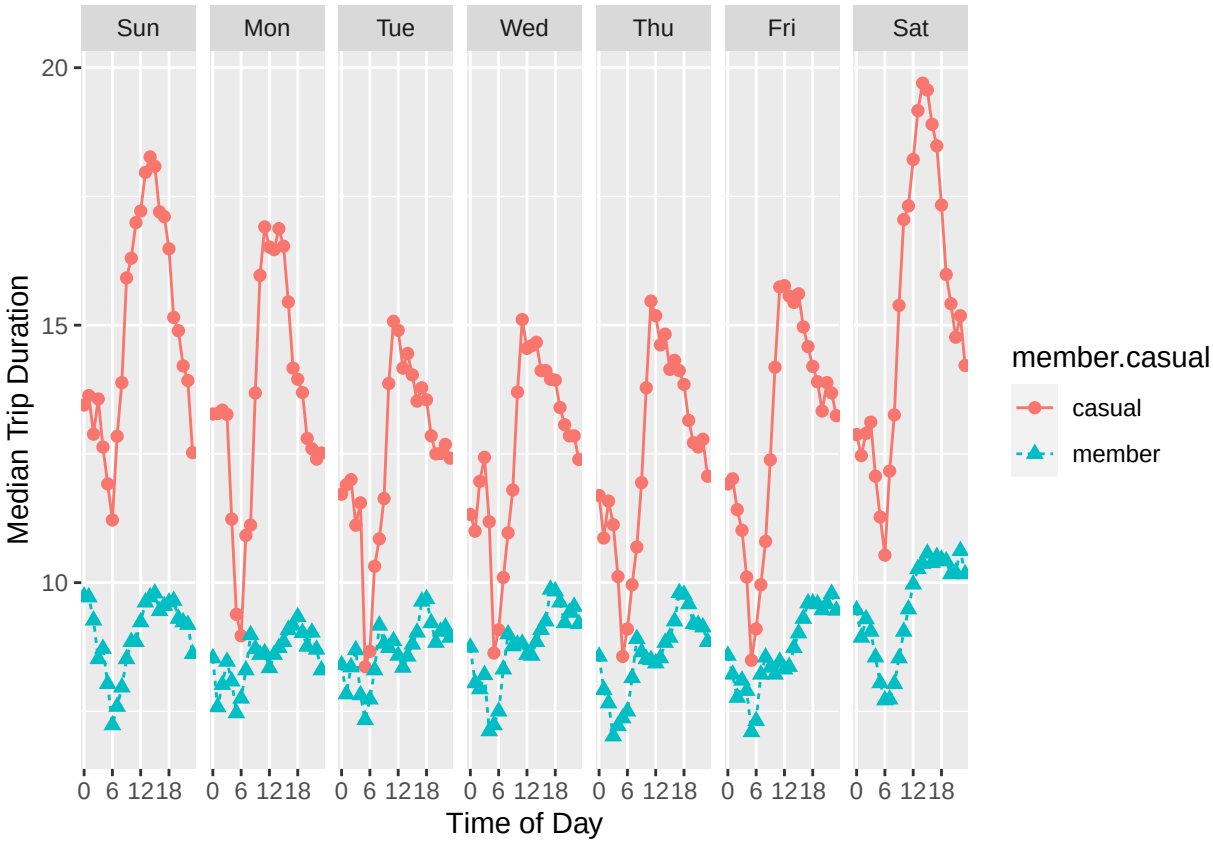
Evaluation of the 'end.station.id' for the NYC based bike trips includes docking stations located in New Jersey. The Citi Bike bikeshare system does include docking stations in Jersey City and Hoboken. A number (XXX) of bike trips end in New Jersey which does remove the bike from the NYC-based area.

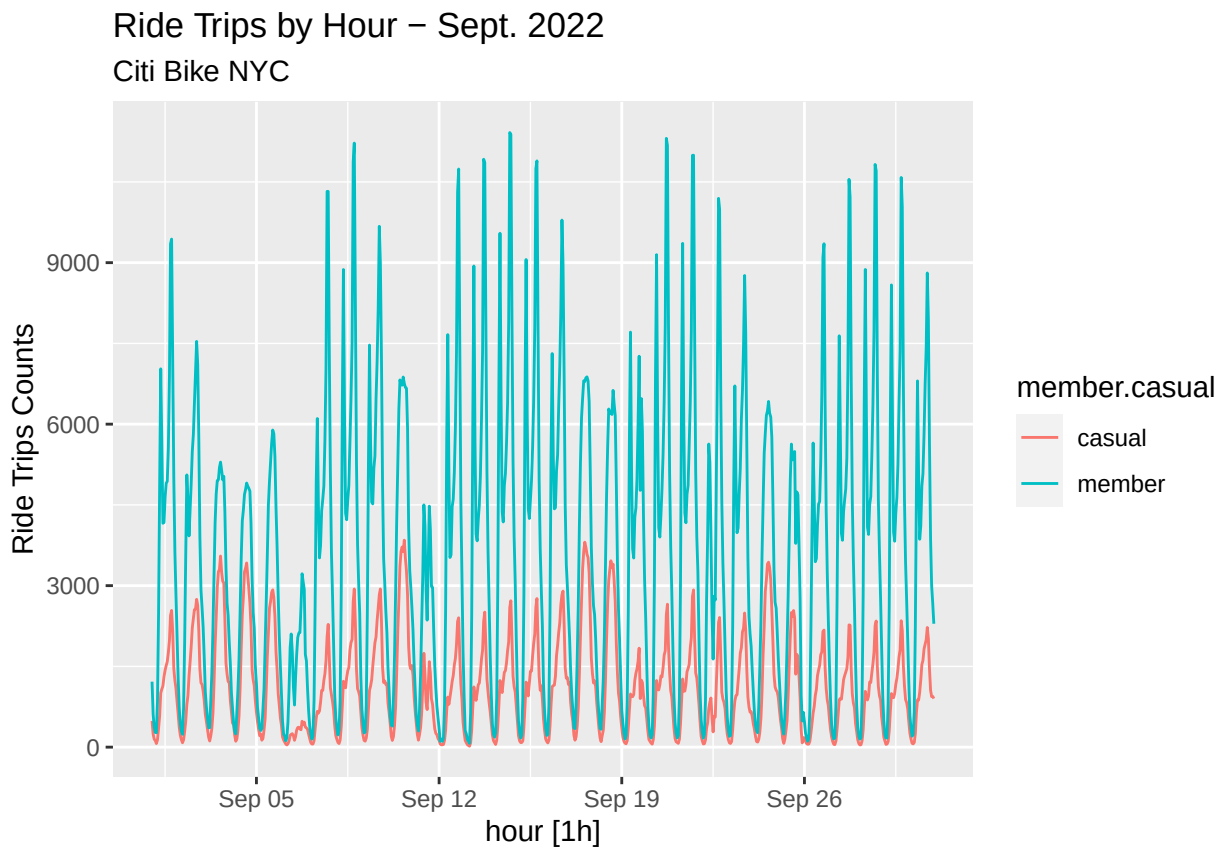
- Building the matrix of surplus by 15 minute interval and station ID
- Overall surplus
- Building the network graph object

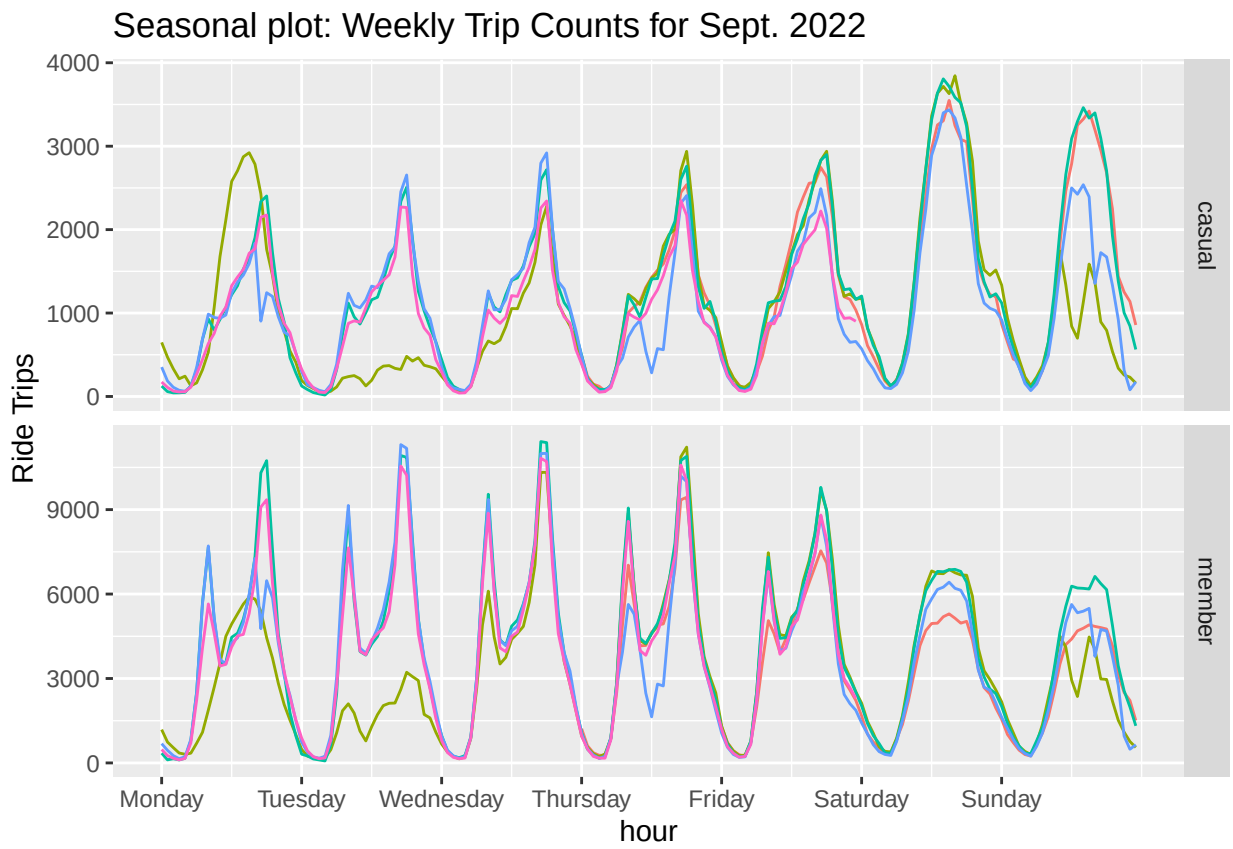
Data Exploration & Analysis




0.06
0.04
0.02
0.00
Number of Trips
0
25
50
75
100
Trip Duration (min.)
member.casual
casual
member

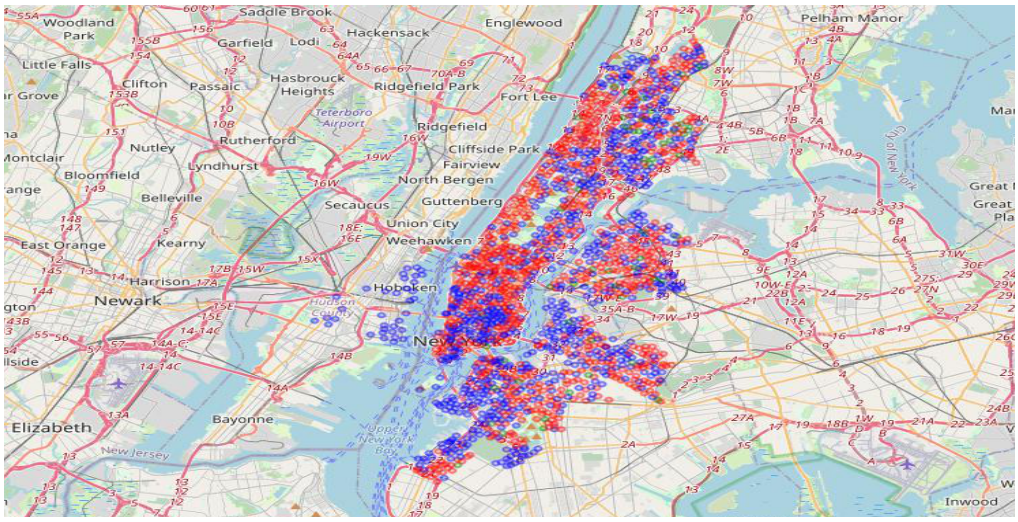

Sun
Mon
Tue
Wed
Thu
Fri
Sat
20
15
10
Median Trip Duration
0 6 12 18
Time of Day
member.casual
casual
member


Ride Trips by Hour − Sept. 2022
Citi Bike NYC
Ride Trips Counts
9000
6000
3000
0
Sep 05
Sep 12
Sep 19
Sep 26
hour [1h]
member.casual
casual
member




```
library(jpeg)
img <- readJPEG("stations_surplus_sum.jpg")
plot(1:10,ty="n", axes = 0, xlab='', ylab='', main='Overall Monthly Surplus by Docking Station')
rasterImage(img,-1,-1,12,12)
```

Overall Monthly Surplus by Docking Station



There are start_station_id and end_station_id that start with 'SYS' with a valid lat/long, so keep them

There are NO starts with JC or HB, but there are ends with JC and HB

For all valid combinations of start to end, there are 472920 combinations

The above table also indicates the frequency of those trips for start-end combinations.

The top 15 entries of the top 20 are the same start and end station ID. so these are casual rides, I have a feeling these 5 are near tourist destinations, perhaps Central Park

471313 in which the start and the end station IDs are NOT the same

Graph object 644->700

```
## $vertices
## + 2/1657 vertices, named, from d028cfc:
## [1] 2821.05 8428.03
##
## $distance
## [1] 4
```

High-Level Exploration

Important to understand the patterns of the bike trips.

Day of the week, time of day, location

Duration of trip

Visuals

'Average Number of Bike Trips by Time of Day and Weekday'

Correlation

Feature Selection

P9: Data Exploration and analysis P10-14: Data Exploration and analysis P14

XXX Model XXX Analysis

Multivariate Regression Analysis P15-P17: Regression Analysis

Explanation of this being count data, and thus going to focus on negative binomial regression, zero-inflated regression, and one more.

Model 1 Approach

Model 2 Approach

Model 3 Approach

Model Selection

Conclusion

What have I learned? (Probably better phrase here)

P17: Conclusion and Next steps

Areas for Future Research

P18: Future research

Bibliography

P19: Bibiography

Data

Literature

P20:Appendix starts

Appendix with Code

```
# Required packages
library(tidyverse)
library(ggplot2)
library(skimr)
library(lubridate)
library(fpp3)
library(assertthat)
library(igraph)
library(ggraph)
library(ggmap)
library(leaflet)
library(rgdal)
library(RColorBrewer)

# Create additional columns of pertinence
# weekday, day of the month, trip duration in minutes, start hour
citibike <- read.csv("data/202209-citibike-tripdata.csv", check.names=TRUE) %>%
  mutate(day = factor(mday(ymd_hms(started_at))),
         start.hour=factor(hour(ymd_hms(started_at))),
         weekday = wday(ymd_hms(started_at),label=TRUE,abbr=TRUE),
         trip.duration = as.numeric(difftime(ended_at,started_at,units="mins")),
         member_casual = factor(member_casual)) %>%
  rename(ride.id=ride_id, rideable.type=rideable_type, started.at=started_at,
         ended.at=ended_at, start.station.name=start_station_name, start.station.id=start_station_id,
         end.station.name=end_station_name, end.station.id=end_station_id, start.lat=start_lat,
         start.lng=start_lng, end.lat=end_lat, end.lng=end_lng, member.casual=member_casual)

# Figure out abandoned and label them
citibike$end.station.name[citibike$end.station.name == ''] <- "Abandoned"
citibike$end.station.id[citibike$end.station.id  == ''] <- "ABAN"

# Output DF if needed
head(citibike)

# Trip by weekday by segment by time of day
citibike %>%
  group_by(day, member.casual, start.hour) %>%
  summarize(n=n(),
            weekday=weekday[1]) %>%
  group_by(weekday, member.casual, start.hour) %>%
  summarize(n.m=mean(n)) %>%
```

```
  ggplot(aes(x=start.hour, y=n.m, fill=weekday)) +
  geom_bar(stat='identity') +
  labs(x='Time of Day',
       y='Number of Trips',
       title='Average Number of Bike Trips by Time, Day, and User') +
  facet_grid(weekday~member.casual) +
  theme(axis.text.x = element_text(size=8, angle=90),
        legend.position = 'none')
# Same as above with density plot instead of histogram
citibike %>%
  filter(trip.duration < 100) %>%
  ggplot(aes(x=trip.duration, fill=member.casual)) +
  geom_density(alpha=0.2) +
  labs(x = 'Trip Duration (min.)',
       y = 'Number of Trips')
citibike %>%
  filter(member.casual %in% c('member', 'casual')) %>%
  group_by(weekday, start.hour, member.casual) %>%
  summarize(med.duration=median(trip.duration)) %>%
  ggplot(aes(x=start.hour, y=med.duration, group=member.casual,
             color=member.casual, linetype=member.casual, shape=member.casual)) +
  geom_point(size=2) +
  geom_line(size=0.5) +
  facet_wrap(~weekday, nrow=1) +
  labs(x='Time of Day',
       y='Median Trip Duration') +
  scale_x_discrete(breaks=c(0,6,12,18))

citibike$started.at.ts <- as_datetime(as.character(citibike$started.at))

citibike_by_hour <- citibike %>%
  mutate(hour=lubridate::floor_date(started.at.ts, "1 hour")) %>%
  group_by(hour, member.casual) %>%
  summarize(cnt=n())
#citibike_by_hour

citibike_by_hour_ts <- citibike_by_hour %>%
  as_tsibble(index=hour, key=c(member.casual))
#citibike_by_hour_ts

autoplot(citibike_by_hour_ts, cnt) +
  labs(title = "Ride Trips by Hour - Sept. 2022",
       subtitle = "Citi Bike NYC",
       y = "Ride Trips Counts")
citibike_by_hour_ts %>%
  gg_season(cnt, period = "week") +
  labs(y = "Ride Trips",
       title = "Seasonal plot: Weekly Trip Counts for Sept. 2022")
# Create table of start to end station IDs
stations_cols <- citibike %>%
  select(start.station.id, end.station.id)
stations_table <- as.data.frame((table(stations_cols)))
```

```r
stations_table <- stations_table %>%
  filter(Freq > 0)

stations_table_order <- stations_table[order(-stations_table$Freq),]

stations_table_dif <- stations_table_order %>%
  filter(as.character(start.station.id) != as.character(end.station.id))

# Create table of start to end station IDs
stations_cols <- citibike %>%
  select(start.station.id, end.station.id)
stations_table <- as.data.frame((table(stations_cols)))

stations_table <- stations_table %>%
  filter(Freq > 0)

stations_table_order <- stations_table[order(-stations_table$Freq),]

stations_table_dif <- stations_table_order %>%
  filter(as.character(start.station.id) != as.character(end.station.id))

stations_table_dif1 <- stations_table_dif
stations_table_dif2 <- stations_table_dif

# Added all=TRUE to account for one-sided counts
stations_table_dif_merge <-
  merge(stations_table_dif1,
      stations_table_dif1,
      by.x=c('start.station.id','end.station.id'),
      by.y=c('end.station.id','start.station.id'), all = TRUE)

# Set the NA (one-sided trips) to count of 0
stations_table_dif_merge[is.na(stations_table_dif_merge)] <- 0

stations_table_dif_merge$surplus <- stations_table_dif_merge$Freq.y - stations_table_dif_merge$Freq.x
stations_table_dif_merge <- stations_table_dif_merge[order(-stations_table_dif_merge$surplus),]

# Remove rows with start station id equal to ABAN for abandoned, those are a result of the merge all=TR
stations_table_dif_merge <- stations_table_dif_merge %>% filter(start.station.id != 'ABAN')

# Want to identify the surplus (or not) by station for the month
station_surplus_count <- stations_table_dif_merge %>%
  group_by(start.station.id) %>%
  summarize(surplus.sum=sum(surplus))


station_surplus_count <- station_surplus_count[order(-station_surplus_count$surplus.sum),]
colnames(station_surplus_count)[1] <- "station.id"

# Extract just the end station Id, lat, log
# because this had the higher count from the initial dataset, going with end_station_id
end_station_info <- citibike %>%
  select(end.station.id, end.lng, end.lat)
```

```
end_station_info <- end_station_info[!duplicated(end_station_info$end.station.id),]

# Now I want the coordinates of all those station_ids
station_surplus_count_coords <- merge(x = station_surplus_count, y = end_station_info, by.x = 'station.

colnames(station_surplus_count_coords)[3] <- "lng"
colnames(station_surplus_count_coords)[4] <- "lat"

station_surplus_count_coords <- station_surplus_count_coords %>%
  add_row(station.id = "ABAN", surplus.sum=1363, lng=-73.99, lat=40.67)

# Using basemaps for NYC
m <- leaflet(data=station_surplus_count_coords) %>%
#  setView(zoom=12) %>%
  addTiles() %>%
  addCircleMarkers(
    ~lng, ~lat,
    popup=~as.character(station.id),
    label=~as.character(station.id),
    radius=.5,
    color = ~ifelse(surplus.sum >= 1, 'blue',
                    ifelse(surplus.sum == 0, 'green', 'red'))
  )

# Display map
#m
library(jpeg)
img <- readJPEG("stations_surplus_sum.jpg")
plot(1:10,ty="n", axes = 0, xlab='', ylab='', main='Overall Monthly Surplus by Docking Station')
rasterImage(img,-1,-1,12,12)
# This removed the duplicate rows by transposing the start and end station ids
stations_table_dif_merge_temp <- stations_table_dif_merge %>% select(start.station.id, end.station.id)


stations_for_graph <- stations_table_dif_merge_temp[!duplicated(lapply(as.data.frame(t(stations_table_di

g_stations <- graph_from_data_frame(stations_for_graph, directed=FALSE, vertices=station_surplus_count_

edges_for_plot <- stations_for_graph %>%
  inner_join(station_surplus_count_coords %>% select(station.id, lng, lat), by=c('start.station.id' = 's
  rename(x=lng, y=lat) %>%
  inner_join(station_surplus_count_coords %>% select(station.id, lng, lat), by=c('end.station.id' = 'sta
  rename(xend=lng, yend=lat)

#assert_that(nrow(edges_for_plot) == nrow(stations_for_graph))

farthest_vertices(g_stations)

# https://yihui.org/en/2018/09/code-appendix/
```

LaTex help https://tex.stackexchange.com/questions/10684/vertical-space-in-lists