Predicting Citi Bike Availability in NYC

DATA 698 Research Project
CUNY Fall 2022

Philip Tanofsky

2022-11-25

v2: started at 11:43A on Friday, Nov. 25, 2022 (final presentation submitted)

Introduction

New York City offers a multitude of transit options including the subway, buses, rideshare, taxis, and since 2013 a bikeshare option known as Citi Bike. The privately owned bikeshare system boasts a total of 25,000 bicycles and over 1,700 stations in New York City, New York in parts of Manhattan, Brooklyn, Queens, and the Bronx along with stations in Jersey City and Hoboken in New Jersey. The bikeshare option provides a solution for the 'first-' and "last-mile" connectivity in urban areas for individuals wanting to access the subway or bus line without walking. Citi Bike offers real-time availability of bicycles through the Citi Bike app, but the availability of a bicycle at the starting location and availability of a docking station upon arrival is not guaranteed at any time. The availability of bicycles is dependent on the riders themselves and the rebalancing strategies of Citi Bike.

The trip data publicly provided by Citi Bike on a monthly basis can be used to examine ridership patterns and directions of flow based on time, date, and type of user (subscriber or casual). This project will attempt to construct a model of the Citi Bike bikeshare system to predict the availability of one or more bicycles based on location and a future date and time.

The Problem

Predict bike availability to avoid an empty docking station

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. Identify system rebalancing and its impact on 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

For this project, I focus on a variety of algorithm model approaches used to predict bikesharing system station availability along with factors that determine the impacts to trip generation volumes. The literature review identifies the negative binomial based model as a common approach. The literature review also finds




other model approaches to assess availability such as LDA, random forest, and neural networks. Finally, I focus the research on bikesharing system rebalancing techniques to better understand how the availability of bikes depends on the movement of bikes separate from user trips.

The literature review focuses primarily on docked, instead of dockless bike-sharing systems across the United States and international locations as the Citi Bike system of New York City is a dock-based system. The project research accounts for model approaches broader than availability prediction to yield a more comprehensive understanding of the strong spatial and temporal nature of the bike-trip data.

Negative-Binomial Based Models

A majority of the literature on bikesharing availability focuses on negative binomial models due to the overdispersion of zeros in the station-level data. The negative binomial models often rely on socio-demographic and weather factors in model construction.

Wang, Lindsey, Schoner, and Harrison (2016) apply log-linear and negative binomial regression models to evaluate the factors of population size, trip attraction and transportation network as primary factors on average daily-station trips. Wang et al. (2016) confirm bikeshare locations near the Minneapolis central business district, college campuses, parks, and bodies of water have higher ridership. Wang et al. (2016) also identify a negative coefficient on the measure of station proximity indicating an over-saturation of bike stations can decrease ridership at the station level.

Hosseinzadeh, Karimpour, and Kluger (2021) assess two micromobility methods, e-scooters and bikesharing, to examine the spatial and temporal factors of ridership using models based on linear regression, negative binomial regression, time series, and generalized additive before focusing on a combined negative binomial generalized additive model to account for the over-dispersion. Hosseinzadeh et al. (2021) identify the amount of rain, existence of thunderstorms, and temperature index as significant factors in ridership. Hosseinzadeh et al. (2021) also find the day of the week along with major holidays and special events impact micromobility use.

Ma, Ji, Jin, Wang, and He (2018) construct an approach based on activity spaces around metro stations which finds the activity space larger on weekdays compared to weekends. Ma et al. (2018) then apply two model types, ordinary least squares (OLS) regression and spatial error model (SEM) with a focus on several

socio-demographic, travel-related, and built environment factors that uncover spatial autocorrelation in the trip data. Ma et al. (2018) observe the SEM outperforms the OLS regression based on model fit with results showing proportion of local residents increases activity size on the weekends while high job-density significantly decreases the activity size on the weekdays leading to a severe imbalance of bike availability on weekdays.

Wang and Chen (2020) explore built-environment, bicycle-infrastructure, and transit-related factors to predict bikeshare station capacity using several models - negative binomial, zero-inflated negative binomial (ZINB), negative binomial-structural equation model (SEM), and finally a combined ZINB-SEM. The integrated approach of ZINB and SEM addresses the impact of excess zeros found at the station level. Wang and Chen (2020) posit the number of docking stations plays a more important role in attracting ridership over factors such as sidewalk length, population density, job density, and areas of green space. Wang and Chen (2020) determine the number of nearby stations, the number of bicycle racks, and bike route length have a greater impact on trip generation at the station level and also observe food services demonstrate an important role in bikeshare use.

Noland, Smart, and Guo (2016) develop trip-generation models that account for proximity of subway stations as a prominent factor. Noland et al. (2016) separate the models by key factors such as day of the week and type of user (subscriber or casual) while controlling for spatial autocorrelation. The results identify clear seasonal and weekly variations in the bikeshare usage. Noland et al. (2016) follow an intuitive approach of dissecting the data to generate several models based on commonalities instead of a single model in order to improve availability forecasting. The results recognize weather and user-base growth as impediments to accurate forecasting.

Alternate Model Approaches

To broaden the understanding of model approaches to bikesharing availability, the literature review expands to research attempting alternate model approaches in evaluating the bikesharing system through Latent Dirichlet allocation (LDA) topic modeling, random forest classification, and multi-layer neural network prediction model.

Li, Huang, and Axhausen (2020) take a different approach for predicting bicycle accessibility by using Dirichlet multinomial regression (DMR) topic modeling based on LDA topic modeling to determine the destination activities for a dockless system while also utilizing a distance decay model. The DMR model predicts the purpose of bike trips based on the points of interest near trip destinations along with arrival times. While the system under consideration is dockless, Li et al. (2020) identify longer trip distances for leisure activities over targeted destinations such as work and school. The model confirms bike accessibility decreases further from the city center and points of interest.

Zhou, Wang, and Li (2019) apply a classification approach to predict the likelihood of an individual selecting bike-sharing or a taxi through spatiotemporal distributions of trip patterns. Zhou et al. (2019) settle on a random forest model based on features including travel distance, time of day, travel directions, weather and land use to predict the mode of transportation. After attempting models based on simple logistic regression and multi-layer neural network, the random forest model proved most valuable based on accuracy and better computational cost.

Lin, He, and Peeta (2018) predict hourly usage at a station level using a graph convolutional neural network with data-driven graph filter (GCNN-DDGF) for adjacency matrices based on spatial distance, demand, average trip duration, and demand correlation. Lin et al. (2018) find the neural network uncovers hidden heterogeneous pairwise correlations between stations labeled as communities to predict the station-level demand without reliance on the weather, socio-demographic, and built environment factors common among most bikesharing system research.

Hyland, Hong, Pinto and Chen (2018) use a hybrid approach combining clustering and regression to model the bikeshare usage. The clustering approach separated the bikeshare stations into clusters based on types of trips (work or leisure) along with proximity to other bikeshare stations. Hyland et al. (2018) then apply

regression algorithms to forecast interactions between the station-clusters via a clustering approach. Hyland et al. (2018) also include other societal and economic data such as labor-force participation, mean income, and population density in the regression models of usage. The clustering of stations by usage type and location provides a blueprint for viewing availability not as a station-by-station model but instead based on clusters of stations formed by physical proximity and usage type.

System Rebalancing

Finally, I wanted to better understand the system rebalancing efforts that directly impact the availability of bikes. Many of the bikesharing systems utilize a form of rebalancing to ensure a greater availability of bikes during high-demand times and high-demand locations. The rebalancing by definition disrupts the natural pattern of user-generated trips.

Qian, Jaller, and Circella (2022) attempt to find an equitable distribution of bikeshare stations through an optimization model defined as a genetic algorithm inspired by natural selection. Qian et al. (2022) find that maximizing revenue through rebalancing results in a smaller network of stations and does not promote accessibility of bikeshare stations to disadvantaged communities. Qian et al. (2022) use the 2016 expansion of Divvy stations in Chicago to evaluate their forecast of additional stations based on two goals: maximizing revenue and improving bikeshare accessibility to disadvantaged communities. Qian et al. (2022) conclude the two goals do not work in concert and thus provide a policy suggestion in which local governments can offer incentives to private companies to improve equitable distribution.

Médard de Chardon, Caruso, and Thomas (2016) focus on existing rebalancing operations through spatial and temporal exploration of the bikeshare trip data across many cities. Medard de Chardon et al. (2016) find rebalancing efforts directly impact the availability of bikes at transit hubs. Medard de Chardon et al. (2016) conclude the aggregation of spatio-temporal trip flow cannot be strictly understood as the spatial demand and natural flow of trips are influenced by system rebalancing. Chardon et al. (2016) also identify induced demand based on rebalancing. Rebalancing strategies are deemed necessary to maintain a high quality of customer service that possibly run counter to the operations defined SLAs, profit increases, or trip maximizations.

The literature review provides a broad perspective of model algorithms for predicting bikesharing system availability at the station level from negative binomial to neural networks. The research also identifies common factors, including day of the week, weather, and socio-demographic information, that directly impact bikesharing use. Finally, the review of system rebalancing provides additional insight into the greater movement of bikes throughout the system which directly impacts availability and trip generation volumes.

Data Collection & Preprocessing

Data Collection

Citi Bike provides individual bike trip data on a monthly basis available at https://ride.citibikenyc.com/system-data. The August through October 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 docking station. 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.

The NYC Open Data (free public data published by New York City agencies and partners) provides a GeoJSON file for the polygons defining each neighborhood 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**

Ridership patterns

- Timeframe: August - October of 2022 (92 days)
- Location: New York City departing trips
- Number of trips: 10,210,102
- Docking Stations: 1,717
- Abandoned trips: 24,887

Bike Availability

- Timeframe: Oct. 31 - Nov. 13 of 2022 (14 days)
- Location: Brooklyn
- Docking Stations: 474

Challenges

Large volume of data

- Over 10 million trips in 3 months
- Over 1700 docking stations in NYC

Rebalancing

- System action of moving bikes to restock docking stations

- Identification and inconsistency

User-friendly availability prediction

- Independent variables to prediction model
- End-user input values

Given the count of over 10 million trips and over 1700 docking stations in NYC, it's quite a large data set which on the one hand makes for we believe some interesting visualizations but on the flip side pushed our computing powers to its system constraints and ultimately was the reason for focusing the prediction model on Brooklyn instead of the entirety of NYC.

Another challenge is accounting for rebalancing performed by Citi Bike staff. Rebalancing is the movement of bikes by the system operators, which are not accounted for the three months of bike share data. For each of the three months, the Citi Bike staff rebalanced over 78000 bikes which equates to an average of over 2000 rebalanced bikes per day. This inability to comprehensively identify rebalanced bikes was the motivation to call the API for accurate availability information.

Final identified challenge. We wanted to build a model to predict bike availability, and make the model user friendly. We envisioned a website or app requiring just a few inputs from a user to then provide the model prediction. This constraint forced us to limit the model inputs to information likely available to a given user.

Data Preprocessing

Upon initial inspection of the 10,210,102 bike trips in September 2022, a total of 24,887 entries do not contain an `end.station.id` and `end.station.name` listed. These 24,887 without a defined destination docking station will be defined as 'Abandoned' meaning the user did not properly 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 of bike trips end in New Jersey which does remove the bike from the NYC-based docking stations of which this research is focused.

Surplus Calculation

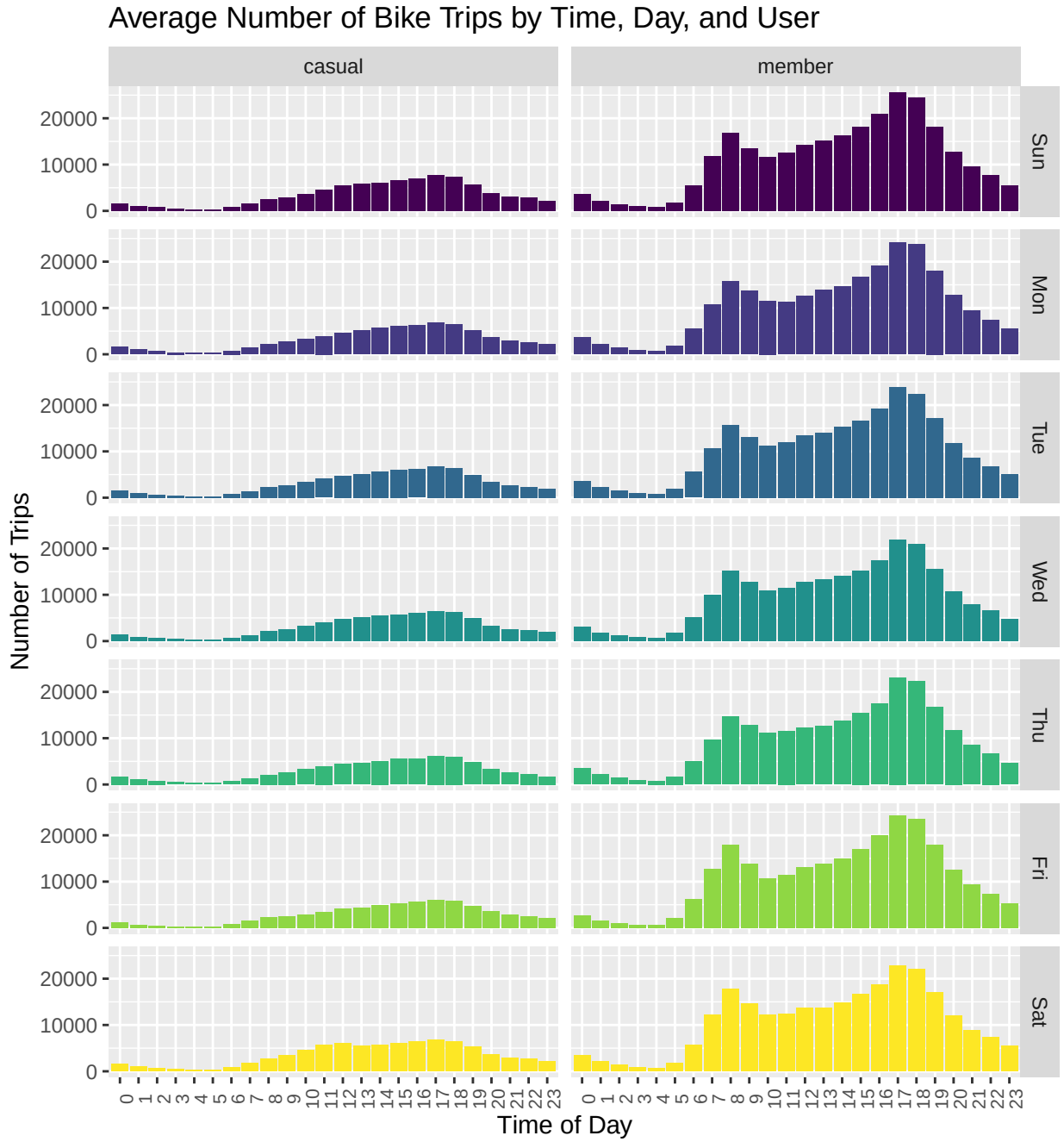
The individual bike trip information was sorted by timestamp and grouped by docking station for each 15-minute interval to count the number of bikes departing and the number bikes arriving. By subtracting the number of departures from the number of arrivals for each station for each interval, we are able to determine the running increase or decrease of bikes at the docking station. This total is defined as the variable `surplus`. A summation of the `surplus` for each docking station is calculated over the course of the month to determine which docking stations are more likely departure stations or arrival stations.

Data Exploration & Analysis

Exploratory data analysis is performed on the Citi Bike trips for August through October 2022 to evaluate the patterns of bike use and identify docking stations with surplus. First, the count of bike trips are assessed to find the high volume days of the week and time of day. Next, the duration of bike trips are evaluated to assess when bikes are individually likely to be unavailable longer. Finally, the surplus of bikes by docking station and borough are analyzed to determine which areas of the New York City are more prone to having lower bike availability.

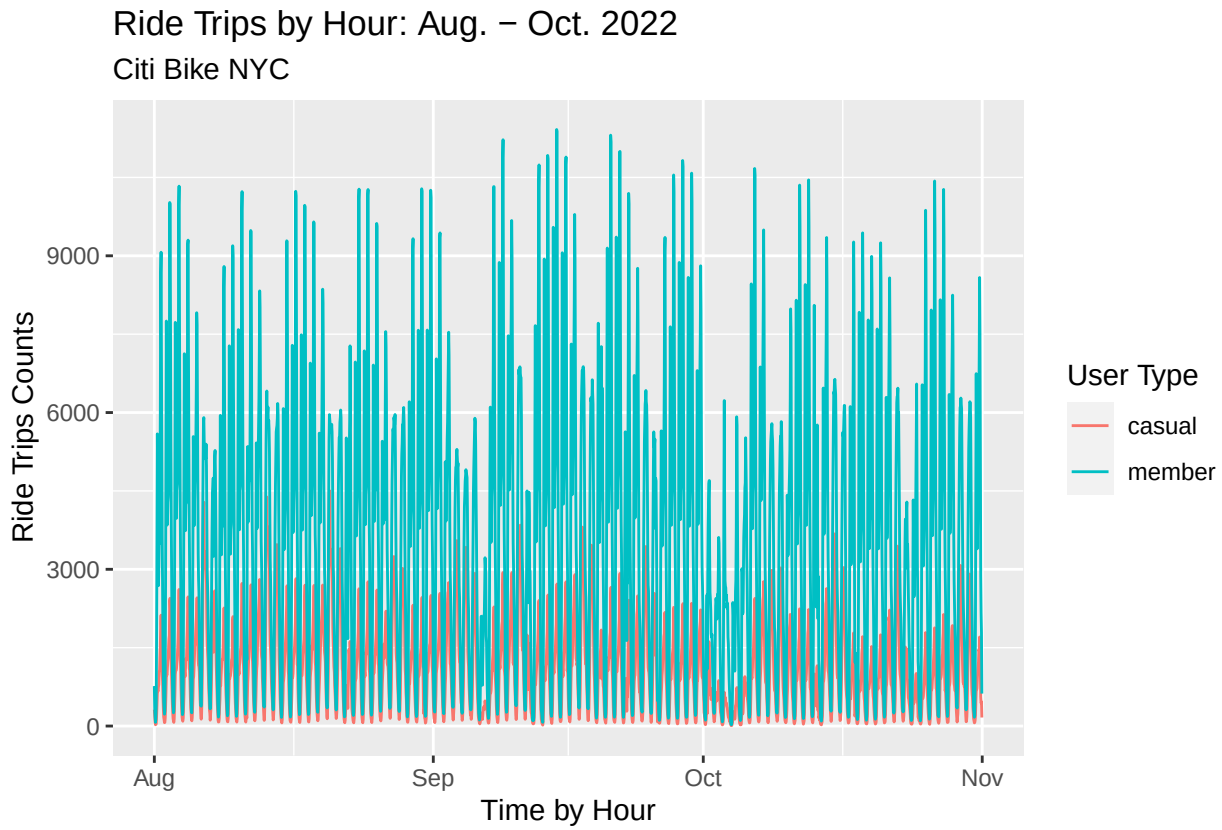
Count of Bike Trips

The count of bike trips are separated by user type - member and casual. With the separation by user type, two distinct patterns emerge of bike use over the course of the day and over the course of a week. The member trips are more likely to follow the workday pattern of spikes in the morning and evening as individuals are traveling to work or from work. The casual users show a pattern not necessarily indicative of the workday but instead of tourist or recreational use. The member trips account for an overall higher volume of trips.

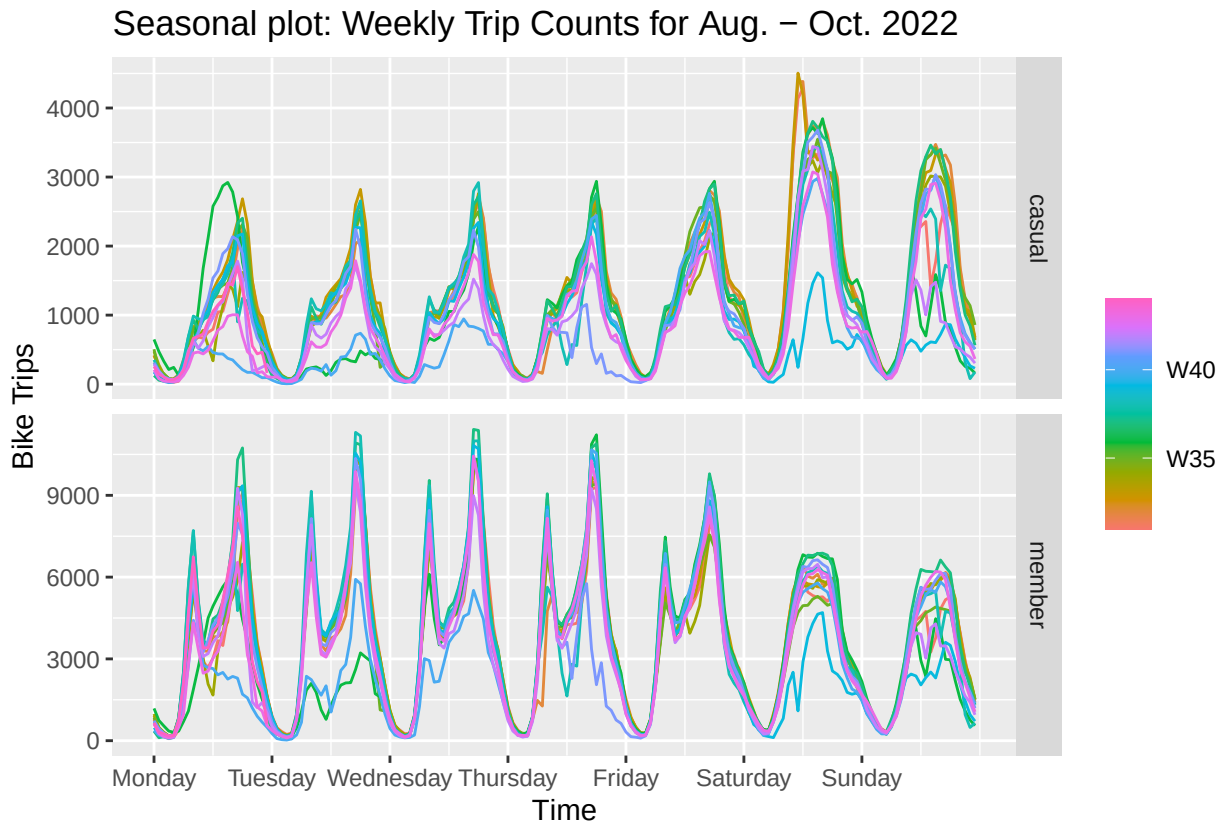


The member user counts show rush-hour spikes for members on weekdays whereas casual users do show higher counts around 5pm and 6pm on weekdays. The comparison of the two user groups also points to greater usage overall by member users. For everyday of the week, the member average member counts per

hour are greater than the casual users. On weekend days, both user groups show a pattern indicative of recreational use with plateau use during the middle of the day and without distinct spikes found on the weekdays. Also, the higher usage of by member users throughout the middle of the day may indicate even if someone is a member the primary reason may not be transportation to and from work.



!!!Words here about the decomposition.



The time-series chart 'Ride Trips by Hour' confirms the higher usage by members with a clear pattern of weekday volumes corresponding to transportation for work purposes. The casual users follow a daily pattern with increases toward the end of the week - Thursday through Saturday.

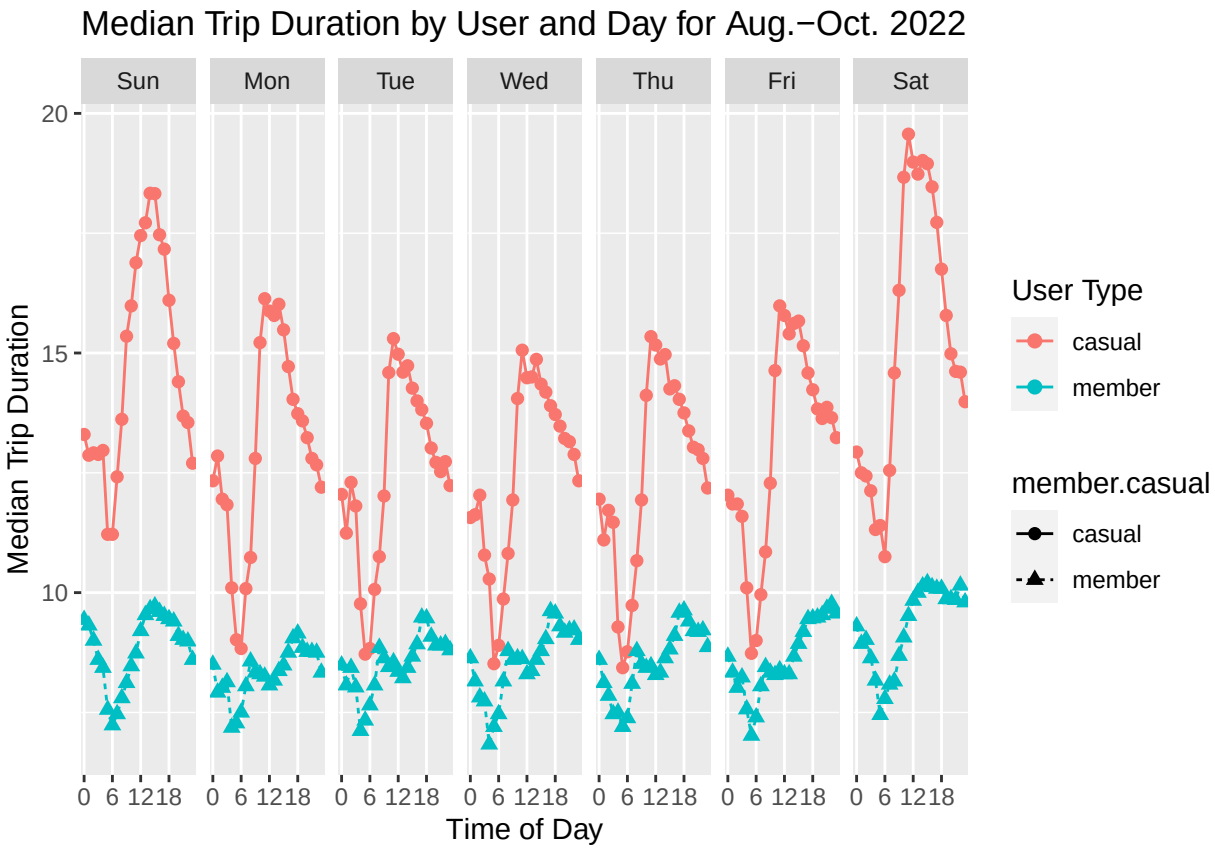
The weekly seasonal plot 'Seasonal plot: Weekly Trip Counts' denotes the same pattern week over week. Based on the plot, the bike trips show consistency from week to week based on member users utilizing the bikes for work and casual users renting the bikes for recreational purposes.

By count of trips per day of the week, Wednesday appears the highest volume day. Given the data comes from September 2022, we believe this observation is a result of pandemic return-to-office policies in which individuals more likely to return to office during the middle of the week instead of Monday or Friday for those with hybrid schedules.

Note: On Tuesday, Sept, 6 2022, the weather consisted of rain the entire day and thus a clear drop in use is evident in both user groups.

Duration of Bike Trips

Next, we evaluate the duration of bike trips to better understand longevity of individual bike unavailability along with reason for trip. As expected, the average bike trip for all users and all times are below 30 minutes as the rental defined time is 30 minutes with users incurring additional fees beyond the base time limit.



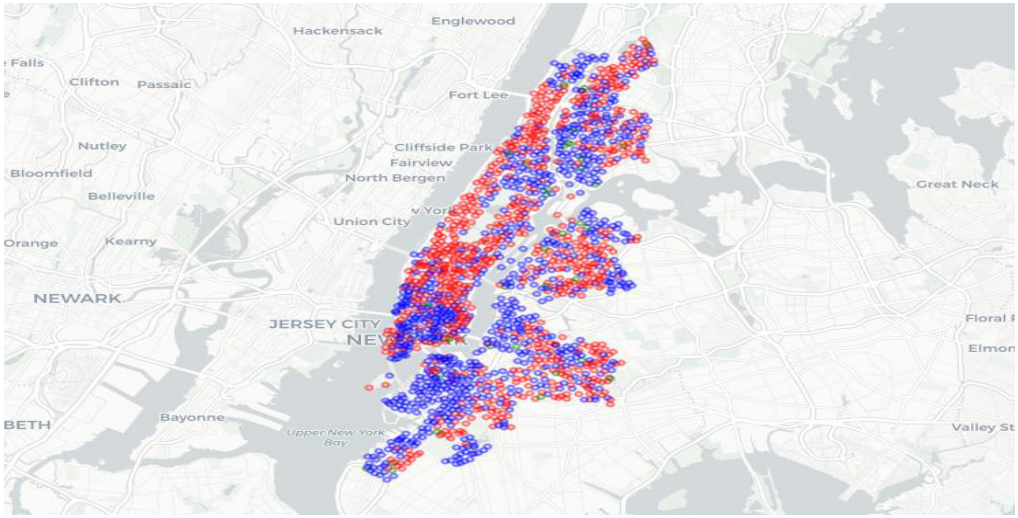
The chart of average trip duration by user type shows a clear distinction between the user types. Member users tend to average bike trips of 10 minutes or less throughout the week, whereas the casual users have a greater variance in average duration based on time of day and day of the week. The member users only average trips greater than 10 minutes during the afternoon and evening on Saturday while casual users typically average even longer trips of greater than 17 minutes on the weekend. Casual users tend to reach of peak of greater than 15 minutes on average everyday of the week, particularly around lunch on weekdays.

We observe that member users tend to have longer average trips during the morning and evening rush hours on weekdays with a slight deviation on Friday evening as the weekend starts. The afternoon and evening hours of Saturday and Sunday show longer trips for members as likely the result of recreational trips.

Docking Station Surplus

After constructing a matrix of bike arrivals and departures for each NYC-based docking station for every 15-minute interval of August through October 2022, we calculate the overall surplus or shortage of bikes. The researchers note, the shortage of bikes can logically not fall below zero without the introduction of rebalancing throughout the bikeshare system. The following analysis confirms the rebalancing of bikes across the docking stations occurs to ensure popular departure docking stations have bikes available despite the dearth of bike trip arrivals.

Overall Monthly Surplus/Shortage by Docking Station



The plot of docking stations across the New York City maps the location of every docking station along with a color to indicate a net positive, negative or even amount of bikes for the given month. The blue stations are net positive, and red stations indicate net negative while green stations are even for the entire month. Of the 1656 docking stations represented, 799 ended with a surplus, 804 with a shortage, and 53 with an even count.

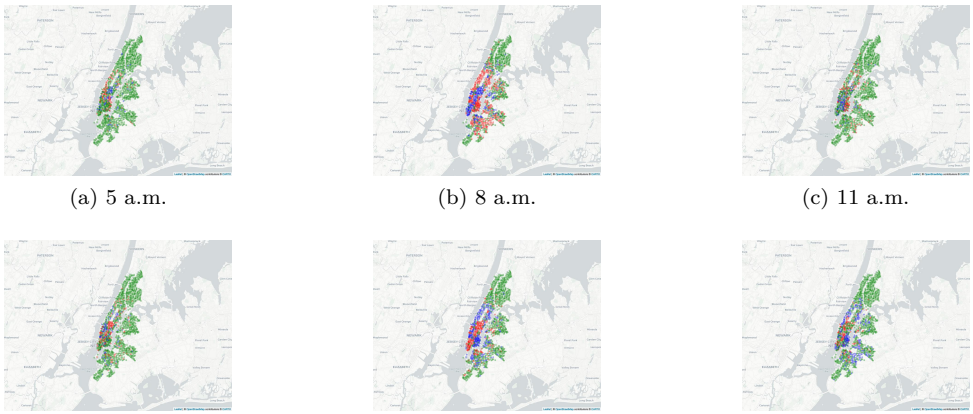
The plot does indicate several blue-colored docking stations in New Jersey. As the dataset does not contain any bike trips originating in NYC, a number of trips have an ending station in Jersey City or Hoboken. The New Jersey based stations are guaranteed to be blue based on that dataset without any departing trips from those docking stations.

The dataset does encompass 472,920 valid combinations of departure and arrival docking stations. Of the valid combinations 15 of the top 20 combinations are the same departure and arrival docking stations, indicative of recreational bike trips. Of the aforementioned valid combinations, 471,313 denote travel between two different docking stations.!!!

EDA: Daily Pattern

Time lapse of surplus or shortage for every NYC docking station

Now, to take a step deeper, we've taken the average arrivals and departures across all Wednesdays, purposely picking a weekday to see the movement throughout the day. Starting, at 5am, mostly green indicating likely little movement of bikes. At 8am we see arrivals in Midtown and the Financial district, two locations of commercial offices while we see red in more residential areas of upper West Side and the East Village. 11am, primarily green as the morning rush has subsided. 2pm, similar to 11am, mostly green. Then at 5pm, we see the opposite of the 8am map. Here we see red, more departures from Midtown and the Financial District will more arrivals in the Upper West Side and East village. 8pm, similar to 5pm we anticipate the movement of bikes away from commercial areas toward residential areas with many green stations through the city.



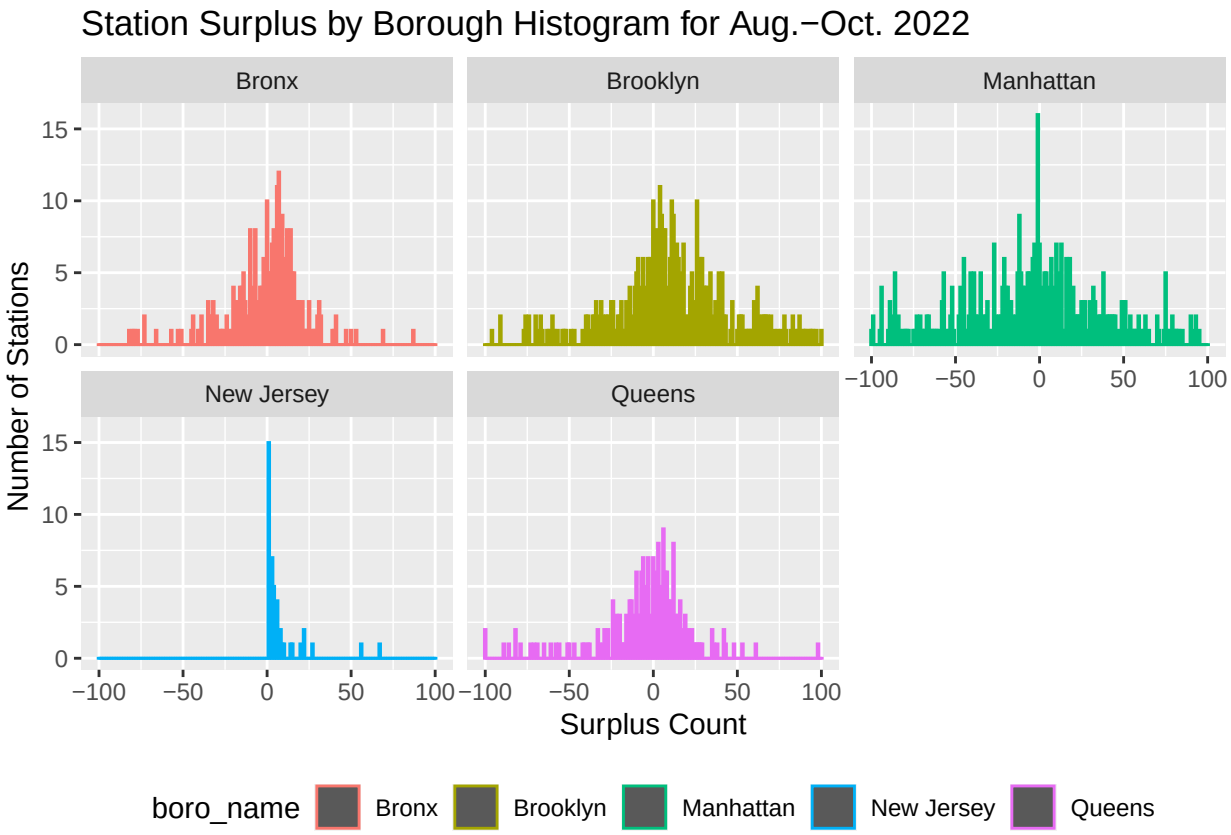
(a) 5 a.m. (b) 8 a.m. (c) 11 a.m.

(d) 2 p.m. (e) 5 p.m. (f) 8 p.m.

Figure 1: Average Wednesday

Surplus by Borough

With almost an even number of docking stations with a net shortage and a net surplus, we evaluate the shortage and surplus by borough.



The plot denotes a near normal distribution for the four boroughs included in the dataset - the Bronx, Brooklyn, Manhattan, and Queens. As noted previously, the dataset contains only trips originating in New York City, so the plot for New Jersey only indicates docking stations surpluses. The balanced distribution across the four NYC boroughs likely demonstrates the bike trips are contained within a borough. With

average duration less than 10 minutes for member users and less than 20 minutes for casual users, the distance traveled for all bike trips is likely less than three miles. The 30-minute base rental rate dictates a limited travel distance which inhibits users from traveling across boroughs via Citi Bike.

Overview of Algorithm Approach

Goal: Predict number of available bikes in Brooklyn within quarter mile

Inputs

- Latitude and longitude (location on map)
- Day of the Week
- Time of Day

Step 1: Hierarchical Clustering

- Cluster all the Brooklyn docking stations
  - Distance: 400m (Quarter mile)

Step 2: Apply Generalized Linear Models

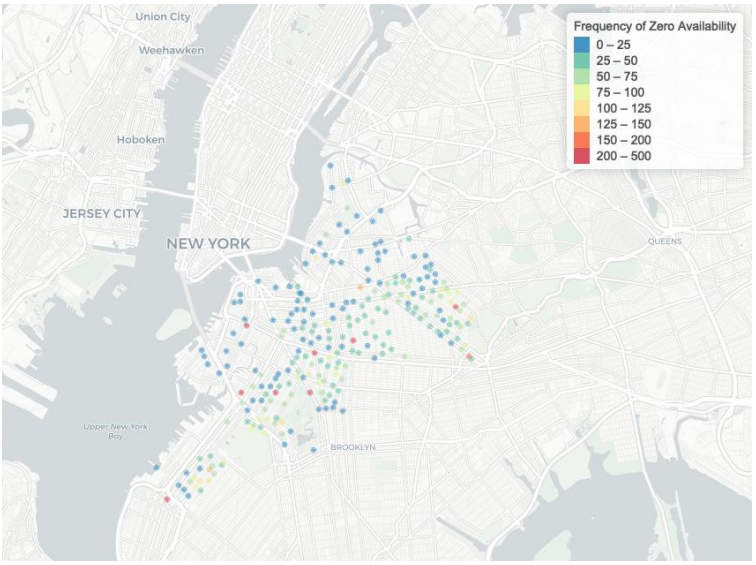
- Predict number of bikes available per cluster
  - Count Models: Poisson, Negative Binomial, Zero-Inflated

The goal of the algorithm is to predict the number of available bikes in Brooklyn within a quarter mile of a given location. The prediction algorithm was scoped to Brooklyn due to the overwhelming amount of data. The 1700 docking stations proved too much given our computer processing constraints. The inputs, again user friendly, are location on a map, day of the week, and time of day. The algorithm is composed of a two step modeling process. First, the 474 Brooklyn docking stations are hierarchically clustered based on a distance of 400 meters, roughly a quarter of a mile. Then, with the availability counts from the API summed to the cluster by time interval, we apply several general linear models or GLMs using count models as a base such as Poisson, Negative Binomial and Zero-inflated to predict the number of available bikes.

Model Decision: Zero Bike Availability

Frequency of zero bikes available per docking station in Brooklyn

- Span: Two weeks
- Instance: 15-minute interval



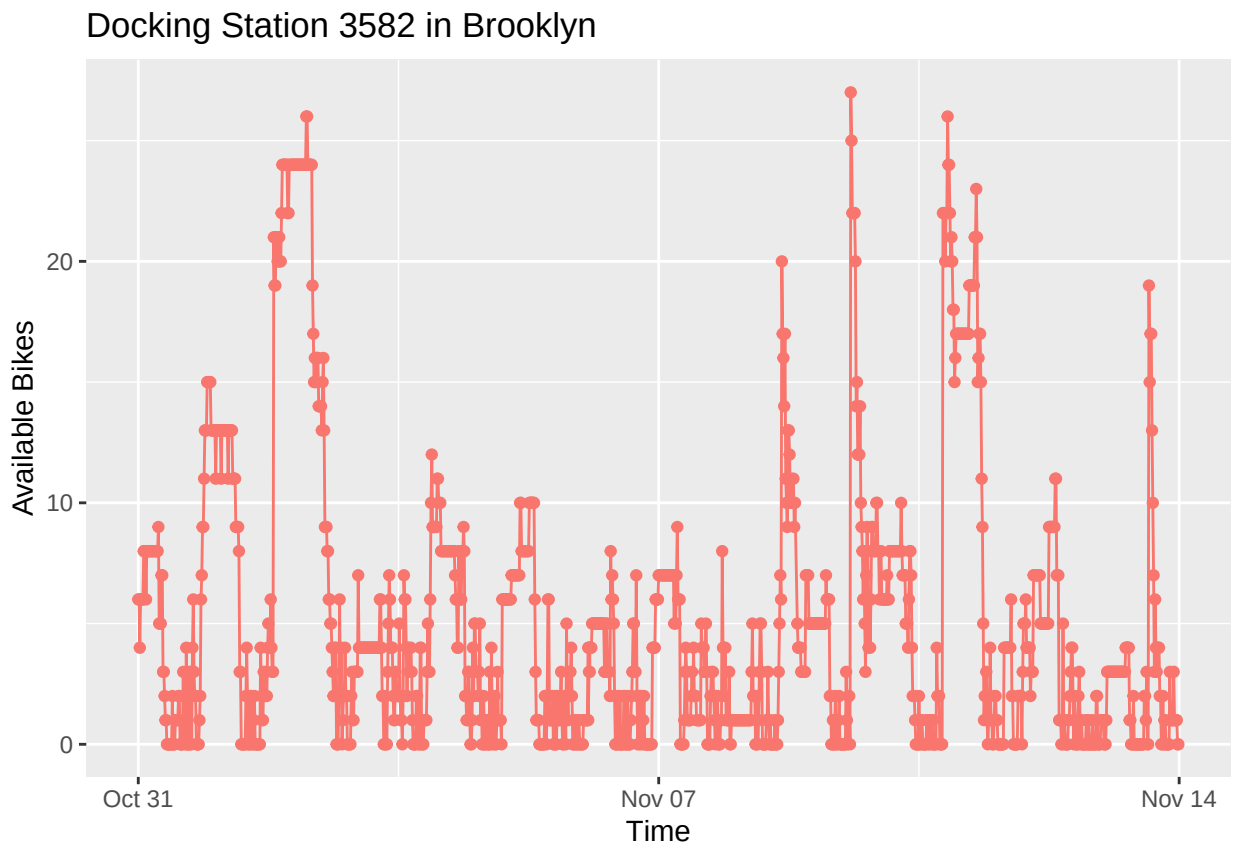
We considered the zero-inflated model algorithm as the count of zero bike availability is common throughout the docking stations of Brooklyn. The map plot shows the docking stations with frequency of zero bike availability. Most docking stations have a lower frequency of 50 or less occurrences across the 1,344 15-minute intervals of the two week period. We do notice a number of red stations indicating high frequency of zero available bikes. These stations could be considered for greater rebalancing. This plot supports the need to consider the zero-inflated model for our purposes of prediction.

Rebalancing Example

Citi Bike reports a rebalancing of 70117 bicycles in September 2022 (https://mot-marketing-whitelabel-prod.s3.amazonaws.com/nyc/September-2022-Citi-Bike-Monthly-Report.pdf).

From report: There were 1625 active stations at the end of the month. The average bike fleet last month was 24,514.0 with 24,327 bikes in the fleet on the last day of the month.

9.6% or almost 10% of bikes rebalanced daily



The above plot shows the occurrence of rebalancing at station 3582 in Brooklyn. Over the course of the two-week period, the sharp spikes which indicate rebalancing are not consistent.

the line chart shows the number of available bikes every 15 minutes for docking station 3582 in Crown Heights Brooklyn. Clearly 4 spikes to 20 or more available bikes occur within the two week span. These sudden increases, or examples of rebalancing, are inconsistent and always return to under 10 available bikes. This chart indicates the departure usage is driven by the available bikes but the arriving bikes remains low during this time span. Given the common yet inconsistent occurrences of rebalancing, we decided to not impute the instances of rebalancing, but instead keep the data as accurate in an attempt at a more valuable model.

Proposed Methodologies

Predictors

- Latitude and longitude converted to cluster
- Day of the Week converted to 'Weekday' or 'Weekend'
- Time of Day in 1-hour intervals
- Average elevation of cluster omitted

Model Methodologies

- Selected: Generalized Linear Model for Counts
  - Poisson Regression Assumptions:
    - Response variable is count per unit of time
    - Independence: Observations are independent in nature

    - Mean equal to Variance: Not true

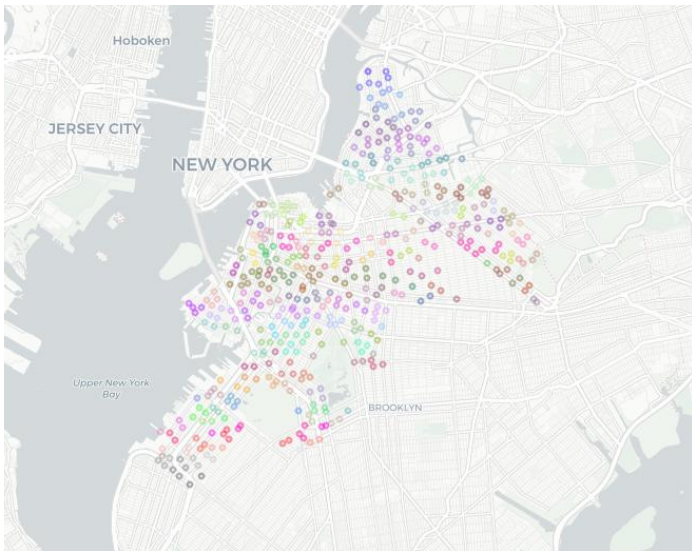
- Alternate Consideration: Time Series Model
    - Forecasting does not provide instance prediction
    - Not easily translatable from user-friendly inputs
    - Requires constantly up-to-date info to forecast from given point in time

For the predictors, we finalized the inputs as longitude and latitude, anticipating the end user wuold actually be providing their GPS coordinates at the time of use. The day of the week is converted to either Weekday or Weekend, in order to simplify the model. The time of day is converted to the start of the hour. Models were constructed using the 15-minute intervals, but the results from the 15 minute intervals showed no improvement compared to the 1-hour intervals. Finally, elevation of the cluster was considered, as anecdotally, stations with higher elevations have fewer returned bikes. The elevation predictor resulted in little significance to the model and thus was omitted.

As for the model algorithms, as mentioned earlier we selected the Generalized Linear Model based on Poisson, Quasi-Poisson, Negative Binomial and Zero-inflated. The availability data meets most of the Poisson regression criteria. Response variable is a count per unit. The observations are independent. Unfortunately, the variance is greater than the mean, which indicates we have overdispersion in the data. Thus the reason we selected Negative Binomial and Quasi-Poisson as base algorithms in addition to Poisson. Also considered, but not used was the time series forecasting model approach. Because the forecasting didn't allow for instance predictions and would have relied on constant up-to-date info, we decided not to include the forecasting approach in the final report.

Modeling Step 1: Clustering Model

- Borough: Brooklyn
- Docking Stations: 474
- Clusters: 213 (2.23 stations/cluster)

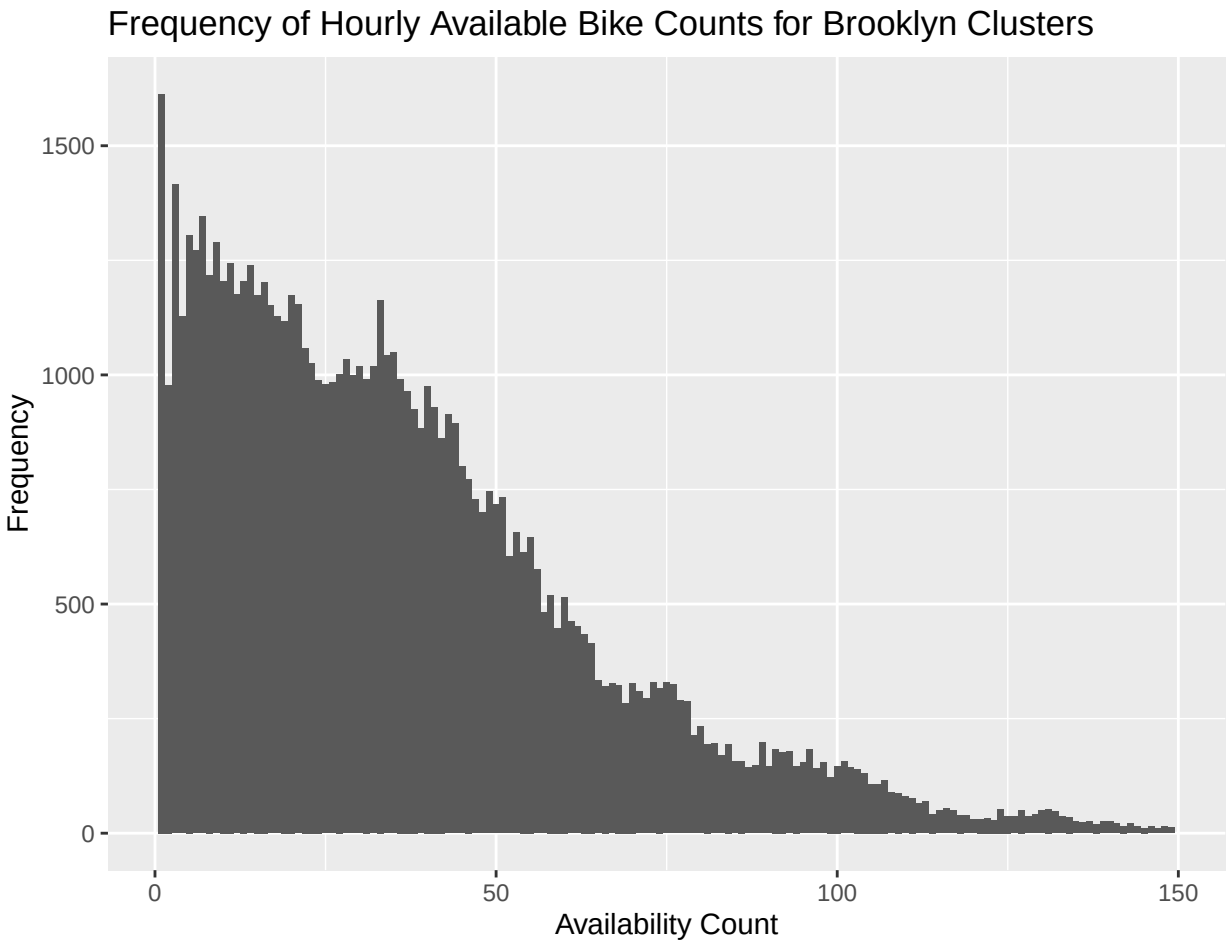


For the clustering, again, we have focused the prediction model on the borough of Brooklyn which contains 474 docking stations. Using the hierarchical clustering from the stats R library, with a distance cutoff of 400 meters (or a quarter of a mile) we end up with 213 clusters or a little over 2 docking stations per cluster. The map plot shows the docking stations colored by cluster number.

Bike Availability Count Frequency

Histogram of availability based on 1HR intervals for Brooklyn clusters

- High frequency at 0 and plateaus at docking station capacities



The histogram shows the frequency of availability counts across all the Brooklyn clusters for each 1-hour interval for the two-week timespan. Here, we see zero is the most frequent value followed by two. Also we see a plateau starting around 25 which marks a station capacity. We see a similar plateau around 60, close to double the single station capacity which would be expected given most clusters are two docking stations. the plot confirms the decision to use the zero-inflated models.

Station Clustering

Filter the stations to NTA name of "Crown Heights North" which results in 11 stations. With over 1650 stations and a walking distance of 400 meters (or one quarter of a mile), the clustering of stations results in over 700 unique clusters. The count of clusters caused vector memory issues in RStudio. In order to continue the research approach, the stations under consideration were decreased to one neighborhood, or NTA, in Brooklyn. The NTA "Crown Heights North" contains 11 docking stations. The clustering of the 11 docking stations based on a distance of 400 meters results in 6 unique clusters, or approximately two docking stations per cluster.

Modeling

Inputs to the models are cluster identifier, day of the week, and time of day. The model inputs would represent a map-based application in which a user selects a point on the map along with day and time to receive an estimation of the bikes available at that future date and time. Based on the user-provided longitude and latitude, a nearest neighbor search is performed to identify the nearest docking station. The translation of the longitude and latitude to the corresponding cluster of the nearest-neighbor docking station would be the input to the model.

For evaluating the models, the two full weeks of data spanning Oct. 31 through Nov. 13 are partitioned into a training set of 80% and a test set of 20%. The dataset follows a time series pattern but in order to predict a specific day and time of the week agnostic of date, regression models for count data are used. The range of regression models for count data considered are Poisson, Quasi-poisson, Negative Binomial, Hurdle, and Zero-Inflated.

The mean of the bikes available by cluster and time interval is 14.0 and the population variance over the dataset is 235.2.

Modeling Step 2: Count Models

Generalized Linear Models for Count Data

5 Models Attempted

- Poisson
  - `stats` library
- Quasi-Poisson
  - `stats` library
- Negative Binomial
  - `MASS` library
- Zero-Inflated
  - Poisson and Negative Binomial
  - `pscl` library

Now, for the heart of the prediction model, we apply 5 distinct models to the availability counts by cluster. First, we use Poisson from the stats library followed by Quasi-Poisson, also from the stats library. Then we attempt Negative Binomial from the MASS library and finally, Zero-inflated, once with Poisson and once with Negative Binomial, relying on the pscl library. We trained on a sample of 75% of the data and tested on the remaining 25% of the data.

Poisson Model The Poisson model is a generalized linear model for count data with the assumption that the variance is equal to the mean.

Explanation the result of the model

Glance explanation

The overdispersion test indicates a dispersion greater than 5 in which true dispersion is defined as 1 or more.

Quasi-Poisson Model The Quasi-Poisson Model is a generalized linear model to model count data, for overdispersed count variable in which the variance is a linear function of the mean.

Model	RMSE	MAE	Missed.Zero
Zero (Poisson)	11.2482	8.0648	280
Poisson	11.3075	8.1143	263
Quasi-Poisson	11.3075	8.1143	263
Zero (Neg. Binomial)	11.8260	8.3998	280
Negative Binomial	12.0320	8.5401	263

Negative Binomial Model The Negative Binomial model is a generalized linear model for count data in which the variance is greater than the mean, and the variance is a quadratic function of the mean.

Zero-Inflated Model A Zero-Inflated Model builds a model for a distribution that allows for frequent zero-values observations. The Zero-Inflated models used in this research use a base of Poisson and Negative Binomial.

Hurdle The Hurdle model is another model for count data consisting of two parts - the probability of attaining a value of zero and a second part to model probability of a non-zero value. The Hurdle models used in this research use a base of Poisson and Negative Binomial.

Model Results

The count model results show the best predictive performance by measuring the Root-Mean-Square Error (RMSE) and Mean Absolute Error (MAE) as the Poisson and Quasi-Poisson models. Overall, the performance of the models falls in a small range with RMSE ranging in 8.68 to 9.42. These results show that the average distance of approximately 9 between the predicted values from the model and the values from the dataset.

To evaluate the 5 models, we made the selection based on the Root-Mean-Square-Error (RMSE) and Mean Average Error (MAE) along with the count of predictions greater than zero when the actual value was zero, noted here as Missed Zero. The Zero-Inflated Poisson based model performed the best according to RMSE and MAE followed by Poisson, Quasi-Poisson, and then Zero Inflated Neg Binomial based model and finally Negative Binomial. Assessing the missed zero column, we notice Poisson, Quasi-poisson and Negative Binomial with the fewest missed zeros at 263 followed by the Zero-Inflated models with 280 missed zeroes each. As the earlier plot of bike availability count frequency by cluster showed, zero was the highest frequency but did not represent an outlier within the frequency distribution. Perhaps the zero-inflated models performed worse at correctly predicting zeroes because the zero frequency wasn't actually inflated as compared to the other values.

In the end we selected the base Poisson model because it resulted in the second best RMSE and MAE and had the fewest missed Zero counts. The Poisson regression equation is presented for the selected model.

Selection: Poisson regression equation

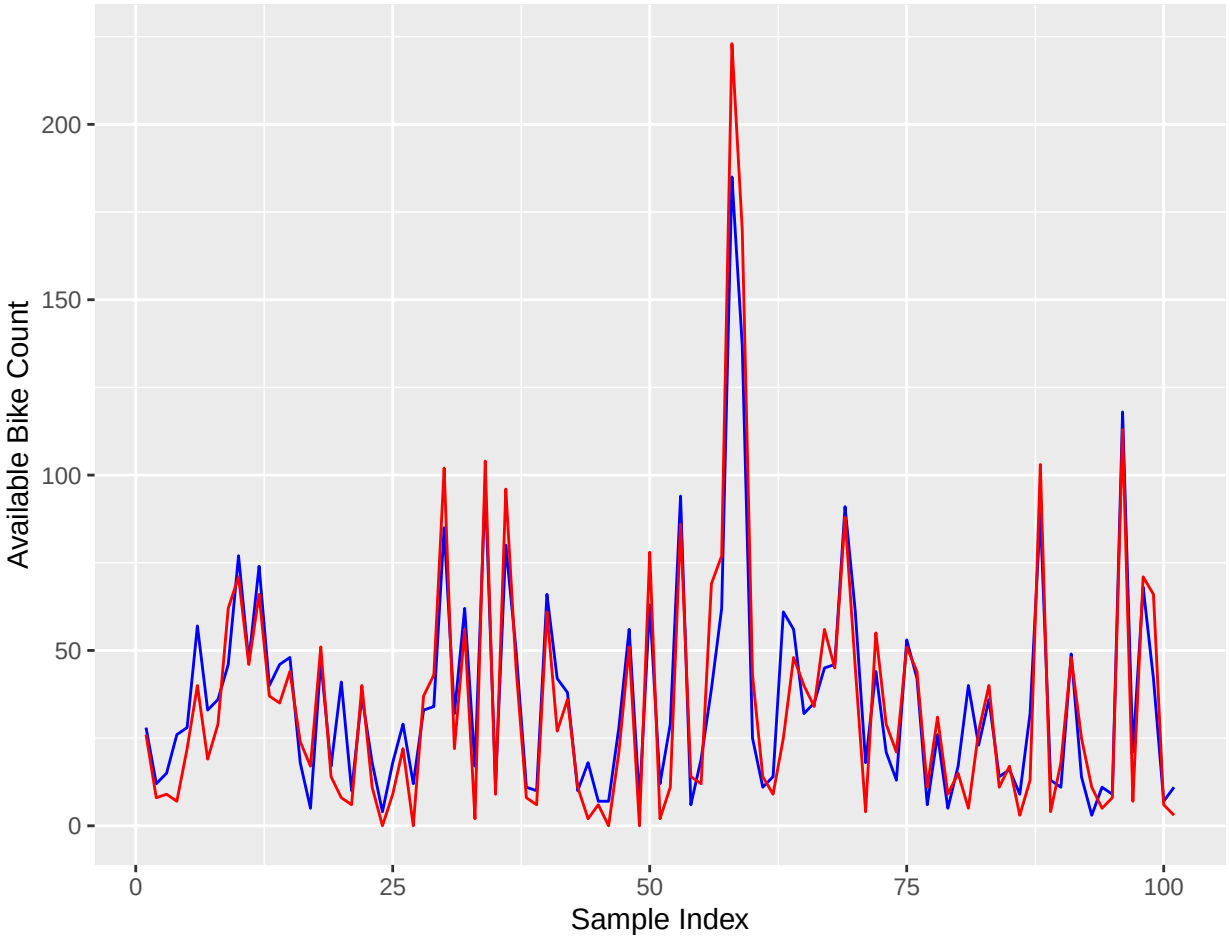
$$log(bikes) = 3.9209 + \beta_1 \cdot cluster + \beta_2 \cdot time + \beta_3 \cdot day \tag{1}$$

Second best RMSE and lowest count of missed Zero availability

Model Results Visualization

Plot subset of Predicted vs Actual counts for select model - Poisson

- Blue: Prediction; Red: Actual



Overall, the 25% test data consisted of over 17000 rows for prediction

The line plot of the prediction subset does show the similar lines of predictions in blue and actual values in Red. We do notice that actual values in Red reach the 0 axis while the blue line does not as often, again indicating the inability of the model to accurately predict the true zero values. Upon visual inspection, the consistency of the two lines does appear close.

Prediction Proof of Concept

To make the model functional, we created a function to predict the number of bikes available given a longitude and latitude, time of day, and day of the week. The longitude and latitude are translated into a cluster identifier based on a nearest neighbor search of the docking stations.

Now, with the prediction model selected and complete. We wrote a function to put the model to practical use as a proof of concept to predict the number of available bikes based on the user-friendly inputs. The output of three trials indicate the number of available bikes and cluster number. The neighborhood provided is displayed to give the user reference of the longitude and latitude entered.

The below example returns a predicted availability of 2 bikes at 2:30PM on Saturday within 400 meters from the corner of Classon Avenue and St. John's Place in Crown Heights Brooklyn.

(-73.960859, 40.67355, "22:00:00", "WDAY")

```
## [1] "Crown Heights - Available Bikes: 13; Cluster: 58"
```

(-73.9617, 40.7192, “18:00:00”, “WKND”)

```
## [1] "Williamsburg - Available Bikes: 57; Cluster: 201"
```

(-74.00509, 40.64338, “17:00:00”, “WDAY”)

```
## [1] "Sunset Park - Available Bikes: 5; Cluster: 10"
```

Current Achievements

Bike Availability prediction model for Brooklyn

- Hierarchical clustering
  - Location-based
- Poisson Model
  - Simplest count model
  - Second best RMSE value
  - Better predictor of zero availability

Ridership usage patterns

- Capture patterns of Citi Bike usage
  - System wide evaluation of NYC
  - Weekly and daily bike trip patterns

We built a bike availability prediction model for Citi Bike docking stations in Brooklyn The model uses hierarchical clustering based on location and a Poisson model for bike count availability based on location, day of the week and time of day. The Poisson model is the simplest count model with the second best RMSE and best predictor of zero availability.

Also, we captured patterns of Citi Bike use across New York City. The system wide evaluation included all five boroughs and clearly denoted weekly and daily bike patterns based on user type.

Future Work

- Deploy web application or smartphone app to use GPS location
- Improve zero availability accuracy
- Model all docking stations of New York City
- Real-time clustering to identify all docking stations within quarter-mile of user at the time
- Increase amount of availability data

We certainly have room to grow and improve. Future work could include deploying the model as a website or smartphone app to actually get the model into users’ hands. Also, the location input could come directly from the phone to make the model truly user-friendly. Next, we should continue to improve the accuracy of the zero availability predictions. A prediction of 10 instead of 8 wouldn’t be the headache to a user the same way a prediction of 2 instead of 0 would be. So to improve the user experience, we should ensure zero availability is predicted accurately with higher frequency.

A few more improvements include a model of all New York City, a real-time clustering to identify all docking stations within a quarter mile of the user at the time and finally, using more availability data. Two weeks is a short window of time, so a larger timeframe we anticipate would improve the model.

Closing Remarks

The exploratory analysis of the Citi Bike dataset for NYC-based trips in September 2022 shows a consistent pattern of use dependent on time of day and day of the week. The pattern persists week over week for the given month. The majority of users are member which indicates the availability of bikes will be dependent on work schedules during weekdays. The surplus and shortage count of bikes by docking station denotes the uneven direction of bikes in some sections of New York City. The shortage (and surplus) counts confirm the practice of rebalancing by Citi Bike to ensure availability of bikes. The duration of bike trips may not play as large a factor in bike availability compared to time of day, day of week, and directional flow of bikes throughout the city.

Links to add: https://ride.citibikenyc.com/system-data https://github.com/MobilityData/gbfs/blob/master/gbfs.md https://ride.citibikenyc.com/system-data/operating-reports

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)
library(jpeg)
library(data.table)
library(MASS)
library(RANN)
library(geosphere)
library(broom)
library(AER)
library(Metrics)
library(pscl)
library(kableExtra)

# Create additional columns of pertinence
# weekday, day of the month, trip duration in minutes, start hour
citibike_202208 <- fread("data/202208-citibike-tripdata.csv", data.table=FALSE, check.names=TRUE)
citibike_202209 <- fread("data/202209-citibike-tripdata.csv", data.table=FALSE, check.names=TRUE)
citibike_202210 <- fread("data/202210-citibike-tripdata.csv", data.table=FALSE, check.names=TRUE)

#citibike <- fread("data/202210-citibike-tripdata.csv", data.table=FALSE, check.names=TRUE) %>%
citibike <- bind_rows(citibike_202208, citibike_202209, citibike_202210) %>%
  mutate(day = factor(mday(ymd_hms(started_at))),
```

```
         start.hour=factor(hour(ymd_hms(started_at))),
         weekday = lubridate::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')

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: Aug. - Oct. 2022",
       subtitle = "Citi Bike NYC",
       x = "Time by Hour",
       y = "Ride Trips Counts") +
  guides(colour = guide_legend(title = "User Type") )
dcmp <- citibike_by_hour_ts %>%
  model(stl = STL(cnt))
components(dcmp) %>% autoplot()
```

```
citibike_by_hour_ts %>%
  gg_season(cnt, period = "week") +
  labs(x = "Time",
       y = "Bike Trips",
       title = "Seasonal plot: Weekly Trip Counts for Aug. - Oct. 2022")
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',
       title = "Median Trip Duration by User and Day for Aug.-Oct. 2022",
       color='User Type') +
  scale_x_discrete(breaks=c(0,6,12,18))
# Create table of start to end station IDs
stations_cols <- citibike %>%
  dplyr::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))

# Create table of start to end station IDs
stations_cols <- citibike %>%
  dplyr::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 %>%
  dplyr::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
img <- readJPEG("stations_surplus_sum.jpg")
plot(1:10,ty="n", axes = 0, xlab='', ylab='', main='Overall Monthly Surplus/Shortage by Docking Station
```

```
rasterImage(img,-1,-1,12,12)
stations_with_elevation <- read.csv('stations_with_elevation.csv', row.names = 1, header= TRUE)

stations_with_boro_hood <- read.csv('stations_with_boro_and_hood.csv', row.names = 1, header= TRUE)

# Combine the elevation, borough, neighborhood, and September surplus
# First let's trim the DF for elevation
stations_with_elevation_trim <- stations_with_elevation %>%
  dplyr::select(short_name, station_id, elevation, elev_units)

stations_with_boro_hood_trim <- stations_with_boro_hood %>%
  dplyr::select(short_name, name, station_id, capacity, ntaname, boro_name, lon, lat)

stations_attrs_trim <-
  merge(stations_with_elevation_trim,
        stations_with_boro_hood_trim,
        by.x=c('short_name'),
        by.y=c('short_name'), all = TRUE)

stations_attrs_trim <- stations_attrs_trim %>%
  dplyr::select(-station_id.y)

colnames(stations_attrs_trim)[colnames(stations_attrs_trim) == 'station_id.x'] <- 'station_id'

stations_attrs_trim[c("boro_name")][is.na(stations_attrs_trim[c("boro_name")])] <- "New Jersey"

stations_attrs_trim <-
  stations_attrs_trim %>%
  mutate(ntaname = ifelse(startsWith(short_name, "JC"), "Jersey City", ntaname))

stations_attrs_trim <-
  stations_attrs_trim %>%
  mutate(ntaname = ifelse(startsWith(short_name, "HB"), "Hoboken", ntaname))


stations_attrs_trim_sur <-
  merge(stations_attrs_trim,
        station_surplus_count,
        by.x=c('short_name'),
        by.y=c('station.id'), all.x = TRUE)

stations_attrs_trim_sur <- stations_attrs_trim_sur %>%
  filter(!is.na(surplus.sum))

# !!! Filter tou New Jersey
stations_attrs_trim_sur %>%
  filter(surplus.sum >= -100 & surplus.sum <= 100) %>%
  ggplot(aes(x=surplus.sum, color=boro_name), ) +
  theme(legend.position="bottom") +
  geom_histogram(bins=201) +
  facet_wrap(~boro_name) +
  labs(x = 'Surplus Count',
       y = 'Number of Stations',
```

```
       title = "Station Surplus by Borough Histogram for Aug.-Oct. 2022",
       fill = "Borough")
# EVAL IS FALSE ... CONSIDER REMOVING!!!
# 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))

citibike_bike_avail <- fread("bike_avail_by_station_and_time.csv", data.table=FALSE, check.names=FALSE)

citibike_bike_avail <- citibike_bike_avail %>%
  dplyr::select(-V1)

citibike_bike_avail_long <- citibike_bike_avail %>%
  pivot_longer(!timestamp, names_to = "station.id", values_to = "bikes.avail")

citibike_bike_avail_long_trim <- citibike_bike_avail_long %>%
  filter(as.integer(station.id) == 3582)

p_rebalance_station <- ggplot(citibike_bike_avail_long_trim,
                              aes(x=timestamp, y=bikes.avail, group=station.id)) +
  geom_line(aes(color=station.id), show.legend = FALSE) +
  geom_point(aes(color=station.id), show.legend = FALSE) +
    labs(x='Time',
         y='Available Bikes',
         title="Docking Station 3582 in Brooklyn")

p_rebalance_station
# Read in the csv of the API responses which have been wrangled together
stations_with_bike_avail <- read.csv('bike_avail_by_station_and_time.csv', row.names = 1, header= TRUE,

stations_with_bike_avail_long <- stations_with_bike_avail %>%
  pivot_longer(!timestamp, names_to = "station.id", values_to = "bikes.avail.count")

stations_with_elevation <- read.csv('stations_with_elevation.csv', row.names = 1, header= TRUE)

stations_with_boro_hood <- read.csv('stations_with_boro_and_hood.csv', row.names = 1, header= TRUE)

# Combine the elevation, borough, neighborhood, and September surplus
# First let's trim the DF for elevation
stations_with_elevation_trim <- stations_with_elevation %>%
  dplyr::select(short_name, station_id, elevation, elev_units)
```

```r
stations_with_boro_hood_trim <- stations_with_boro_hood %>%
  dplyr::select(short_name, name, station_id, capacity, ntaname, boro_name, lon, lat)

stations_attrs_trim <-
  merge(stations_with_elevation_trim,
        stations_with_boro_hood_trim,
        by.x=c('short_name'),
        by.y=c('short_name'), all = TRUE)

stations_attrs_trim <- stations_attrs_trim %>%
  dplyr::select(-station_id.y)

colnames(stations_attrs_trim)[colnames(stations_attrs_trim) == 'station_id.x'] <- 'station_id'

stations_attrs_trim[c("boro_name")][is.na(stations_attrs_trim[c("boro_name")])] <- "New Jersey"

stations_attrs_trim <-
  stations_attrs_trim %>%
  mutate(ntaname = ifelse(startsWith(short_name, "JC"), "Jersey City", ntaname))

stations_attrs_trim <-
  stations_attrs_trim %>%
  mutate(ntaname = ifelse(startsWith(short_name, "HB"), "Hoboken", ntaname))

stations_attrs_trim <- stations_attrs_trim %>%
  rename(short.name=short_name, station.id=station_id, elev.units=elev_units, nta.name=ntaname, boro.na

stations_attrs_trim_bk <- stations_attrs_trim %>%
  filter(boro.name == "Brooklyn")

# Distance matrix for docking stations
# Result is in meters
dist_mat <- distm(stations_attrs_trim_bk[9:10], stations_attrs_trim_bk[9:10], fun=distHaversine)
dist_mat <- as.data.frame(dist_mat)

dist_mat[is.na(dist_mat)] <- 0
dMat <- as.dist(dist_mat)

# Now for the clustering
hier_clust <- hclust(dMat, method = "complete")
# 400 meters is about a quarter of a mile (0.248548 miles)
stations_attrs_trim_bk$cluster <- cutree(hier_clust, h=400)

station_to_cluster <- stations_attrs_trim_bk %>%
  dplyr::select(station.id, cluster, elevation, capacity)

# Merge long df with counts with station.id and cluster to label clusters properly
stations_bike_avail_by_time <-
  merge(stations_with_bike_avail_long,
        station_to_cluster,
        by.x=c('station.id'),
        by.y=c('station.id'), all.x = TRUE)
```

```r
stations_bike_avail_by_time_no_na <- stations_bike_avail_by_time %>%
  filter(!is.na(cluster))

stations_bike_avail_by_time_no_na_group_sum <- stations_bike_avail_by_time_no_na %>%
  group_by(timestamp, cluster) %>%
  summarize_at(vars(bikes.avail.count, capacity), sum)

stations_bike_avail_by_time_no_na_group_mean <- stations_bike_avail_by_time_no_na %>%
  group_by(timestamp, cluster) %>%
  summarise_at(vars(elevation), mean)

stations_bike_avail_by_time_no_na_group <- cbind(stations_bike_avail_by_time_no_na_group_sum, stations_b

# mutate to convert timestamp into day of the week, etc.
stations_bike_avail_by_time_no_na_group <- stations_bike_avail_by_time_no_na_group %>%
  mutate(time=as.ITime(ymd_hms(timestamp)),
         weekday = lubridate::wday(ymd_hms(timestamp),label=TRUE,abbr=TRUE))

stations_bike_avail_by_time_no_na_group$cluster <- as.factor(stations_bike_avail_by_time_no_na_group$clu
stations_bike_avail_by_time_no_na_group$time <- as.factor(stations_bike_avail_by_time_no_na_group$time)
stations_bike_avail_by_time_no_na_group$weekday <- factor(stations_bike_avail_by_time_no_na_group$weekda

stations_bike_avail_by_time_no_timestamp <- subset(stations_bike_avail_by_time_no_na_group, select=-c(ti

stations_bike_avail_by_time_1H <- stations_bike_avail_by_time_no_timestamp %>%
  filter(endsWith(as.character(time), "00:00"))

stations_bike_avail_by_time_1H <- stations_bike_avail_by_time_1H %>%
  mutate(day = ifelse(weekday %in% c('Sat', 'Sun'), 'WKND', 'WDAY'))


# Table of availability based on 1HR intervals
tab_1H <- table(stations_bike_avail_by_time_1H$bikes.avail.count)

# This plot shows the available bike count frequencies given 1HR intervals and Brooklyn station cluster

# There appears to be a max around 22 or so as the distribution drops off just before 25, likely a resu

tab_1H_df <- as.data.frame(tab_1H)

tab_1H_df$Var1 <- as.integer(tab_1H_df$Var1)

# Distribution of count results
p_distro_avail_counts <- tab_1H_df %>%
  filter(Var1 < 150) %>%
  ggplot(aes(x=Var1, y=Freq)) +
  geom_bar(stat="identity") +
  labs(x='Availability Count',
       y='Frequency',
       title='Frequency of Hourly Available Bike Counts for Brooklyn Clusters')

# Display plot
p_distro_avail_counts
```

```r
stations_bike_avail_by_time_1H_imp <- stations_bike_avail_by_time_1H


#71568 rows total in the train set
#2992 in which the bikes.avail.count greater than capacity
#125 clusters have avail greater than capacity

# Impute the available bike count to match the capacity
stations_bike_avail_by_time_1H_imp$bikes.avail.count <- ifelse(stations_bike_avail_by_time_1H_imp$bikes
                                                               stations_bike_avail_by_time_1H_imp$capacity,
                                                       stations_bike_avail_by_time_1H_imp$capacity,
                                                       stations_bike_avail_by_time_1H_imp$bikes.avail.co
#stations_bike_avail_by_time_1H <- stations_bike_avail_by_time_no_timestamp %>%
#  filter(endsWith(as.character(time), "00:00"))

# Start with data partition here.
set.seed(8675309)
index <- sample(2, nrow(stations_bike_avail_by_time_1H_imp), replace = TRUE, p=c(0.75, 0.25))
#index <- sample(2, nrow(stations_bike_avail_by_time_1H), replace = TRUE, p=c(0.75, 0.25))
train_1H <- stations_bike_avail_by_time_1H_imp[index==1,]
test_1H <- stations_bike_avail_by_time_1H_imp[index==2,]

# Sample Variance is 313
#var(stations_bike_avail_by_time_1H$bikes.avail.count)
n <- length(stations_bike_avail_by_time_1H_imp)
# Population variance
#var(stations_bike_avail_by_time_1H$bikes.avail.count)*(n-1)/n
# Mean is 13
#mean(stations_bike_avail_by_time_1H$bikes.avail.count)

# Over dispersion exists
mod_1H_pois <- glm(bikes.avail.count ~ cluster + time + day, data = train_1H, family ="poisson")
#summary(mod_1H_pois)

#dispersiontest(mod_1H_pois)

# Taking the floor as that is the actual available bikes, no partial bikes
pred_1H_pois <- predict.glm(mod_1H_pois, newdata = test_1H, type = "response")
rmse_mod_1H_pois <- ModelMetrics::rmse(test_1H$bikes.avail.count,floor(pred_1H_pois))
mae_mod_1H_pois <- mae(test_1H$bikes.avail.count,floor(pred_1H_pois))
#library(broom)
#glance(mod1)
#library(AER)
#dispersiontest(mod1)
# This indicates dispersion
mod_1H_qp <- glm(bikes.avail.count ~ cluster + time + day, data = train_1H, family ="quasipoisson")
#summary(mod_1H_qp)

pred_1H_qp <- predict.glm(mod_1H_qp, newdata=test_1H, type = "response")
rmse_mod_1H_qp <- ModelMetrics::rmse(test_1H$bikes.avail.count,floor(pred_1H_qp))
mae_mod_1H_qp <- mae(test_1H$bikes.avail.count,floor(pred_1H_qp))
mod_1H_nb <- glm.nb(bikes.avail.count ~ cluster + time + day, data=train_1H)
#summary(mod_1H_nb)
```

```
pred_1H_nb <- predict.glm(mod_1H_nb, newdata=test_1H, type = "response")
rmse_mod_1H_nb <- ModelMetrics::rmse(test_1H$bikes.avail.count,floor(pred_1H_nb))
mae_mod_1H_nb <- mae(test_1H$bikes.avail.count,floor(pred_1H_nb))
mod_1H_zero_pois <- zeroinfl(bikes.avail.count ~ cluster + time + day, data=train_1H, dist = "poisson")
#summary(mod_1H_zero_pois)

pred_1H_zero_pois <- predict(mod_1H_zero_pois, newdata=test_1H,type = "response")
rmse_mod_1H_zero_pois <- ModelMetrics::rmse(test_1H$bikes.avail.count,floor(pred_1H_zero_pois))
mae_mod_1H_zero_pois <- mae(test_1H$bikes.avail.count,floor(pred_1H_zero_pois))
mod_1H_zero_nb <- zeroinfl(bikes.avail.count ~ cluster + time + day, data=train_1H,dist = "negbin")
#summary(mod_1H_zero_nb)

pred_1H_zero_nb <- predict(mod_1H_zero_nb, newdata=test_1H ,type = "response")
rmse_mod_1H_zero_nb <- ModelMetrics::rmse(test_1H$bikes.avail.count,floor(pred_1H_zero_nb))
mae_mod_1H_zero_nb <- mae(test_1H$bikes.avail.count,floor(pred_1H_zero_nb))
# Hurdle
#mod_1H_hurd_pois <- hurdle(bikes.avail.count ~ cluster + time + day, data=train_1H, dist = "poisson")
#summary(mod_1H_hurd_pois)

#pred_1H_hurd_pois <- predict(mod_1H_hurd_pois, newdata=test_1H, type = "response")
#rmse_mod_1H_hurd_pois < -ModelMetrics::rmse(test_1H$bikes.avail.count,floor(pred_1H_hurd_pois))
#mae_mod_1H_hurd_pois <- mae(test_1H$bikes.avail.count,floor(pred_1H_hurd_pois))
rmse_1H <- c(rmse_mod_1H_pois, rmse_mod_1H_qp, rmse_mod_1H_nb,
#             rmse_mod_1H_hur_pois, rmse_mod_1H_hu_nb,
             rmse_mod_1H_zero_pois, rmse_mod_1H_zero_nb)
# Calculate normalized RMSE with 246 as the max bike avail count
rmse_1H_norm <- rmse_1H/246
mae_1H <- c(mae_mod_1H_pois, mae_mod_1H_qp, mae_mod_1H_nb,
#            mae_mod_1H_hur_pois, mae_mod_1H_qp,
            mae_mod_1H_zero_pois, mae_mod_1H_zero_nb)

aic_1H <- c(mod_1H_pois$aic, mod_1H_qp$aic, mod_1H_nb$aic,
#            mae_mod_1H_hur_pois, mae_mod_1H_qp,
            mod_1H_zero_pois$loglik, mod_1H_zero_nb$loglik)

models_1H <- c("Poisson","Quasi-Poisson","Negative Binomial",
#               "h_pois","h_nb",
               "Zero (Poisson)","Zero (Neg. Binomial)")

#data.frame(models_1H, rmse_1H, mae_1H)%>%
#  arrange(rmse_1H)
df_for_plot <- as.data.frame(cbind(test_1H,
                                   floor(pred_1H_pois),
                                   floor(pred_1H_qp),
                                   floor(pred_1H_nb),
                                   floor(pred_1H_zero_pois),
                                   floor(pred_1H_zero_nb))) %>%
  rename(Actual='bikes.avail.count', Pois.Prediction=8, QPois.Prediction=9,
         NB.Prediction=10, Zero.Pois.Prediction=11, Zero.NB.Prediction=12)

df_for_plot$Actual <- as.integer(df_for_plot$Actual)
df_for_plot$Pois.Prediction <- as.integer(df_for_plot$Pois.Prediction)
df_for_plot$QPois.Prediction <- as.integer(df_for_plot$QPois.Prediction)
```

```r
df_for_plot$NB.Prediction <- as.integer(df_for_plot$NB.Prediction)
df_for_plot$Zero.Pois.Prediction <- as.integer(df_for_plot$Zero.Pois.Prediction)
df_for_plot$Zero.NB.Prediction <- as.integer(df_for_plot$Zero.NB.Prediction)

miss_zero_pois <- df_for_plot %>%
  filter(Pois.Prediction > 0 & Actual == 0) %>%
  count()
miss_zero_qp <- df_for_plot %>%
  filter(QPois.Prediction > 0 & Actual == 0) %>%
  count()
miss_zero_nb <- df_for_plot %>%
  filter(NB.Prediction > 0 & Actual == 0) %>%
  count()
miss_zero_z_p <- df_for_plot %>%
  filter(Zero.Pois.Prediction > 0 & Actual == 0) %>%
  count()
miss_zero_z_nb <- df_for_plot %>%
  filter(Zero.NB.Prediction > 0 & Actual == 0) %>%
  count()

miss_zero_1H <- c(miss_zero_pois[1,1], miss_zero_qp[1,1], miss_zero_nb[1,1],
            miss_zero_z_p[1,1], miss_zero_z_nb[1,1])
#RMSE.Norm=rmse_1H_norm
data.frame(Model=models_1H, RMSE=rmse_1H, MAE=mae_1H, Missed.Zero=miss_zero_1H) %>%
  arrange(rmse_1H) %>% kable(digits = 4) %>%
  kable_paper("hover", full_width = F)
# Modify the number of rows to consider
# 17715 rows in DF
df_for_plot_trim <- df_for_plot[500:600,]

p_pred_vs_actual <- ggplot() +
  geom_line(data = df_for_plot_trim, aes(x = 1:nrow(df_for_plot_trim), y = Pois.Prediction), color = "bl
  geom_line(data = df_for_plot_trim, aes(x = 1:nrow(df_for_plot_trim), y = Actual), color = "red") +
  xlab('Sample Index') +
  ylab('Available Bike Count')
#!!! Title needed here

p_pred_vs_actual
predict_num_bikes_avail <- function(longitude, latitude, time, day) {}
predict_num_bikes_avail <- function(longitude, latitude, time, day) {
  #longitude: -73.960859 (in Brooklyn within quarter mile of docking station)
  #latitude: 40.67355 (in Brooklyn within quarter mile of docking station)
  #time: "00:00:00"|"01:00:00"|"02:00:00"|"03:00:00"|"04:00:00"|"05:00:00"|
  #      "06:00:00"|"07:00:00"|"08:00:00"|"09:00:00"|"10:00:00"|"11:00:00"|
  #      "12:00:00"|"13:00:00"|"14:00:00"|"15:00:00"|"16:00:00"|"17:00:00"|
  #      "18:00:00"|"19:00:00"|"20:00:00"|"21:00:00"|"22:00:00"|"23:00:00"|
  #day: "WKND"|"WDAY"

  # Convert lon/lat to cluster
  location <- data.frame(matrix(nrow = 1,data = c(longitude, latitude)))
  closest_station <- nn2(stations_attrs_trim_bk[, 9:10], query=location, k=1)
  inp_cluster <- stations_attrs_trim_bk[closest_station$nn.idx,]$cluster
```

```
  #print(inp_cluster)

  # Create input for prediction
  bike_avail_query <- data.frame(matrix(nrow = 1, data = c(inp_cluster, 0, time, day)))
  colnames(bike_avail_query) <- c("cluster", "bikes.avail.count", "time", "day")

  # Predict using Model: mod1
  num_bikes_avail <- predict.glm(mod_1H_pois, newdata = bike_avail_query, type = "response")
  # Take floor of prediction to ensure whole number
  num_bikes_avail <- as.integer(floor(num_bikes_avail))

  return(c(num_bikes_avail, inp_cluster))
}
# Crown Heights
prediction_CH <- predict_num_bikes_avail(-73.960859, 40.67355,
                        "22:00:00", "WDAY")

print(paste0("Crown Heights - Available Bikes: ", prediction_CH[1], "; Cluster: ", prediction_CH[2]))
# Williamsburg
prediction02_WB <- predict_num_bikes_avail(-73.9617, 40.7192,
                        "18:00:00", "WKND")

print(paste0("Williamsburg - Available Bikes: ", prediction02_WB[1], "; Cluster: ", prediction02_WB[2]))
# Sunset Park
prediction02_SP <- predict_num_bikes_avail(-74.00509, 40.64338,
                        "17:00:00", "WDAY")

print(paste0("Sunset Park - Available Bikes: ", prediction02_SP[1], "; Cluster: ", prediction02_SP[2]))
# https://yihui.org/en/2018/09/code-appendix/
```

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