Predicting Citi Bike Availability in NYC

DATA 698 Research Project: v3 11/26 5:45P

Philip Tanofsky

2022-11-27

Contents

Abstract

P1

P2

P3

P1

P2

P3

P1

P2

P3

P2

P3

Key words: ***Bikeshare, Count Data, Poisson, Negative Binomial, Zero-Inflated***

Empty Citi Bike docking stations in Brooklyn, November 2022

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. Paul Bailo and Alex Ng with support and guidance throughout the process and Evi Wiley for a lifetime of support.

The Problem

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 (member or casual). This project will attempt to construct a model of the Citi Bike bikeshare system to predict the number of available bikes within a quarter mile based on location and a future date and time.

1. Determine the flow of bicycles throughout the Citi Bike system in New York City to identify neighborhoods with a surplus or shortage of Cit bike 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, and location?

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.

Methodology

At stated previously, our research attempts to answer the following questions:

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, and location?

Our methodology for answering these questions follows the steps listed below: Data Collection & Preprocessing, Data Exploration & Analysis, Hierarchical Clustering and Count Data Regression Modeling. The nature of each is described below:

Data Collection & Preprocessing: The data collection followed two separate paths from Citi Bike operators. First, the trip data for three months was downloaded and used to create a timetable for the surplus or shortage of every docking station. Second, the bike availability for every docking station was retrieved from an API based on a running script to construct a timetable of bikes available every 15 minutes across a two-week period.

Data Exploration & Analysis: The bike trip data was used to understand the pattern of Citi Bike usage across New York City and comprehend the patterns across days of the week and dependent on time of day. The analysis of bike trips uncovered the system rebalancing but could not be accurately identify the daily updates thus forcing the reliance on the two weeks of availability data. The exploration also identified the overwhelming number of docking stations across the NYC system thus placing constraints on the proposed prediction model.

Hierarchical Clustering: To fulfill the goal of a prediction model based on location, clustering was performed on the docking stations of Brooklyn in order to decrease the size of the model and also provide a better real-world approach to end-user bike availability.

Count Data Regression Modeling: Identification and evaluation of count data regression models. We constructed several predictive models for count data to identify the most accurate model. The best model for bike availability prediction was selected based on RMSE, MAE, and zero count error rate. The count data regression models were evaluated based on the train-test split approach. The model performances were assessed based on the training and test model metrics.

Based on the selected count data regression model, a function was written to provide true bike availability predictions.

Data Collection & Preprocessing

Data Collection

This project relies on two primary data sources from Citi Bike NYC. The first dataset, the complete set of bike trips for the three months spanning August-October 2022 is used to identify the patterns of bike use across

New York City. The second dataset, an aggregation of docking station status for every 15 minutes for the two-week period of Monday, Oct. 31, 2022 through Sunday, Nov. 13, 2022. The station status API provides real-time bike availability count of every docking station in the Citi Bike system. This two-week dataset provides an accurate depiction of the available bikes, specifically accounting for operational rebalancing.

Citi Bike provides individual bike trip data on a monthly basis available at https://ride.citibikenyc.com/system-data. This project uses three months of bike trip data from August through October 2022 for New York City. 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 station status API (https://github.com/MobilityData/gbfs/blob/master/gbfs.md) contains the bike availability counts for every docking station in the Citi Bike system. The API contains the following pertinent variables for this research project. The total number of available bikes is the sum of `num.ebikes.available` and `num.bikes.available`.

- **num.ebikes.available:** Number of electronic bikes available
- **station.id:** Unique station identifier
- **station.status:** Status of station as "active" or "out of service"
- **num.bikes.available:** Number of classic bikes available

The station information API, also available from Citi Bike, provided the following pertinent variables for each docking station in the system.

- **short.name:** Unique station name (numerical)
- **lat:** Latitude of the docking station
- **lon:** Longitude of the docking station
- **station.id:** Unique station identifier
- **capacity:** Capacity of the docking station

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 to 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

Challenges

We faced three primary challenges during this research project. First, the large volume of data proved overwhelming for a system processing specifications. Given the count of over 10 million trips and over 1700 docking stations in NYC the large data provided a base for several interesting data visualizations but pushed our computing powers to their system constraints and ultimately was the reason for focusing the prediction model on the borough of Brooklyn instead of the entirety of NYC.

The second challenge was the ability for account for system rebalancing performed by the 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 78,000 bikes which equates to an average of over 2,000 rebalanced bikes per day. This inability to comprehensively identify rebalanced bikes was the motivation to call the API for accurate availability information.

The final challenge was self-induced and a driver for the few independent variables used for the prediction model. We wanted to build a user-friendly model to predict bike availability. 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 August through October 2022, a total of 24,887 entries did 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. As a note, Citi Bike does charge a fine for bikes not properly returned to a docking station.

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 across the three months to count the number of bikes departing and the number bikes arriving per interval. 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 three months to determine which docking stations are more likely departure stations or arrival stations.

Availability To prepare the bike availability counts by docking station, we constructed a data table of the 1,344 API request made every 15 minutes over the course of the two weeks, and summing the available classic bikes with electronic bikes and then joining by `station.id` with the station information dataset to align the docking station capacity with the availability numbers. The API reported several docking stations with more available bikes than capacity of the docking station. In these instances, the availability was decreased to match the total capacity of the docking station.

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 a surplus or a shortage. 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.

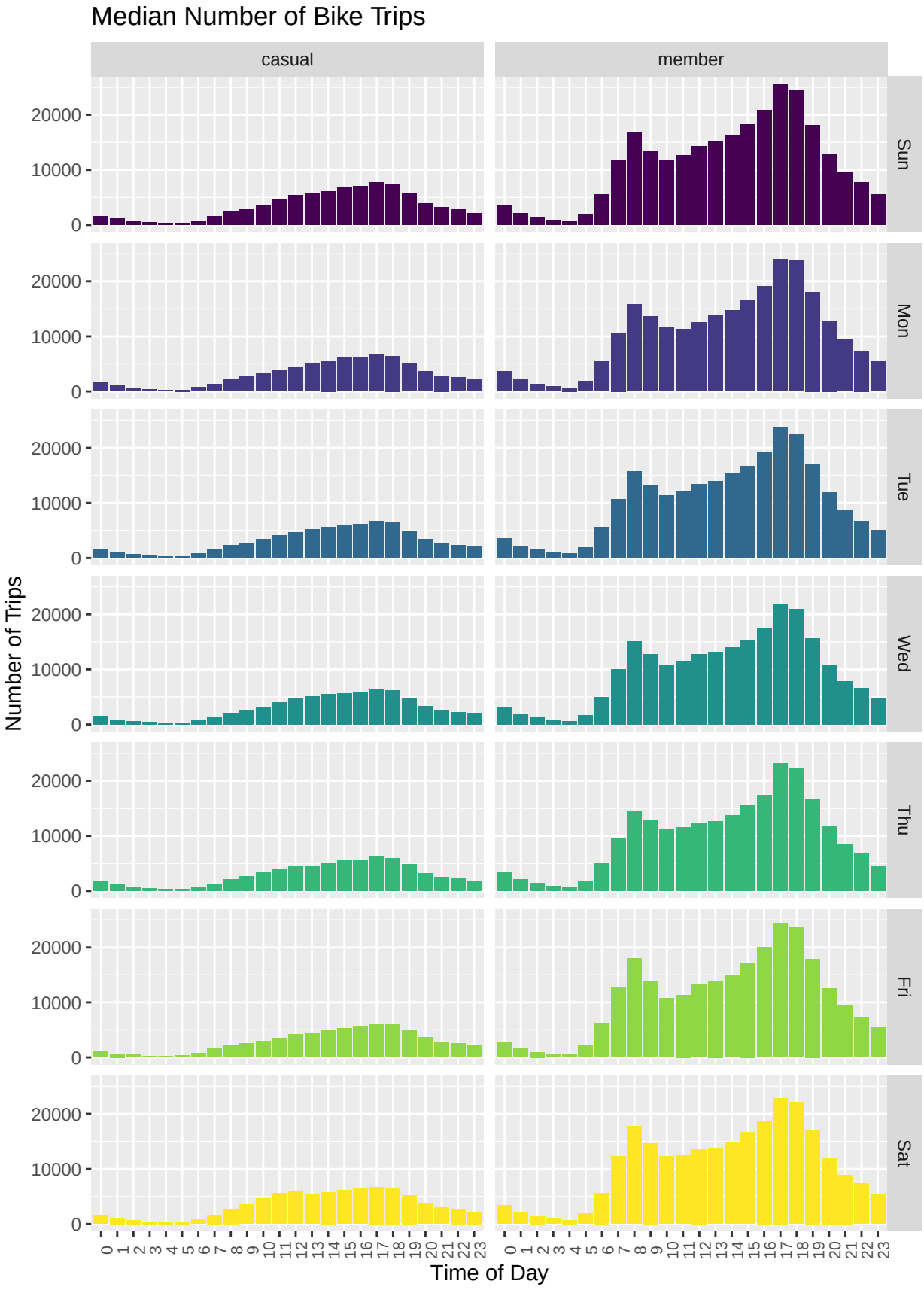
For understanding the ridership patterns of the Citi Bike system, the following statistics outline the breadth of the bike trip information assessed.

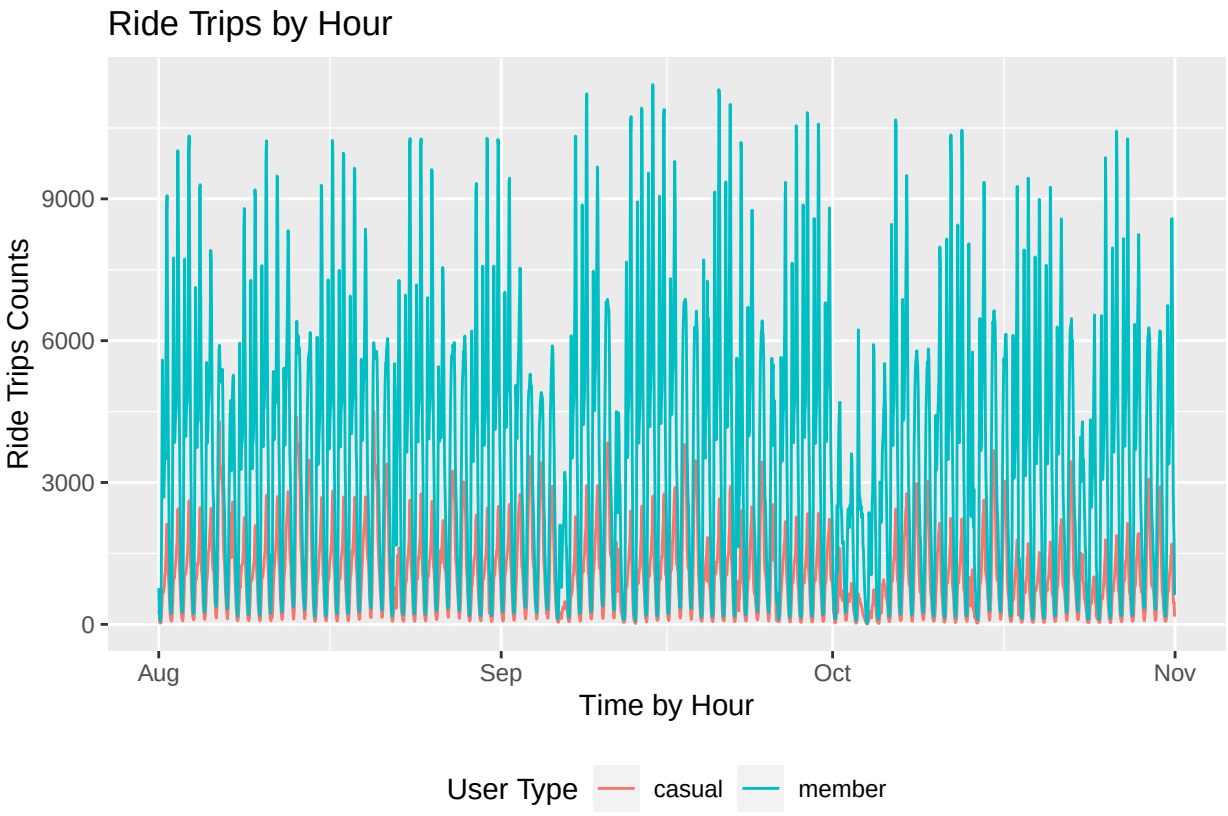
- **Timeframe:** August - October of 2022 (92 days)
- **Location:** New York City departing trips
- **Total trips:** 10,210,102
- **Total docking stations:** 1,717
- **Abandoned trips:** 24,887

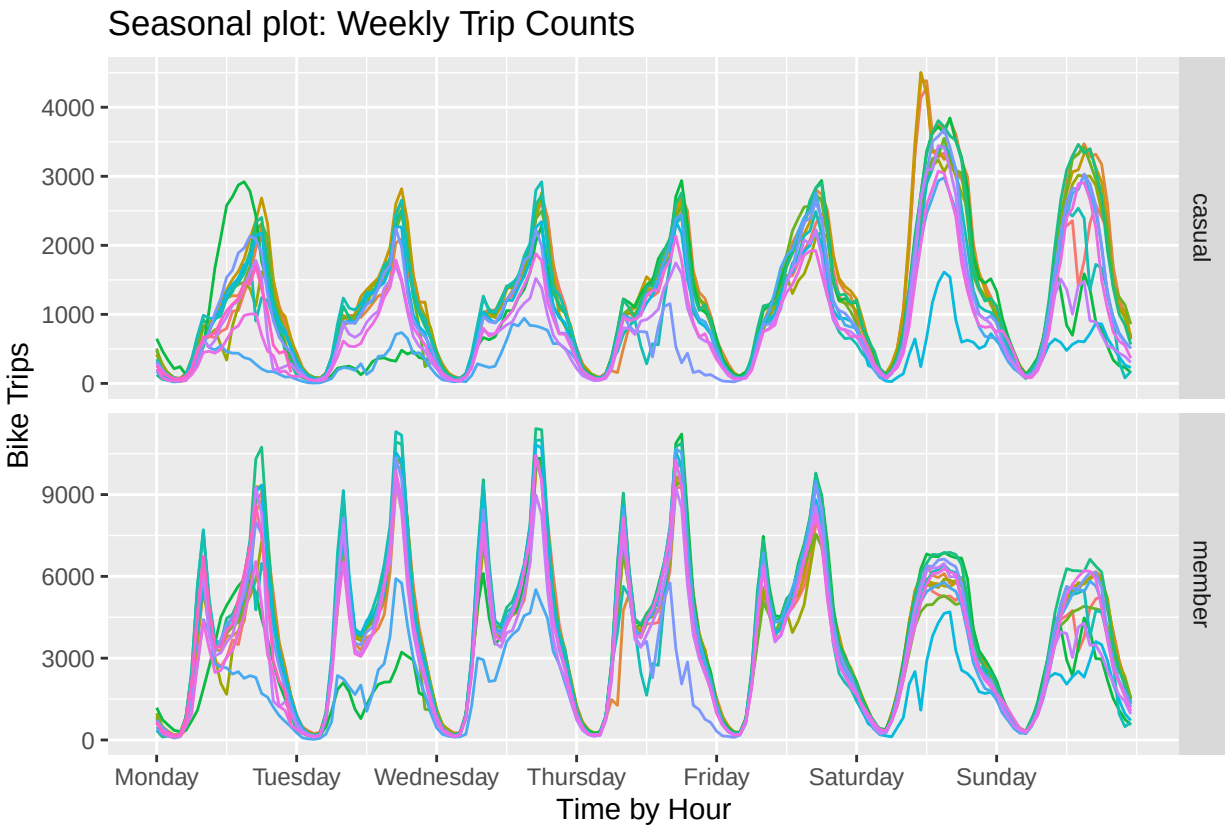
Count of Bike Trips

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 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 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 members For everyday of the week, the 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 a 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.


Median Number of Bike Trips
casual
member
Sun
Mon
Tue
Wed
Thu
Fri
Sat
Number of Trips
0
10000
20000
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Time of Day


Ride Trips by Hour
Ride Trips Counts
9000
6000
3000
0
Aug
Sep
Oct
Nov
Time by Hour
User Type
casual
member


Seasonal plot: Weekly Trip Counts
Bike Trips
4000
3000
2000
1000
0
casual
9000
6000
3000
0
member
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday
Time by Hour

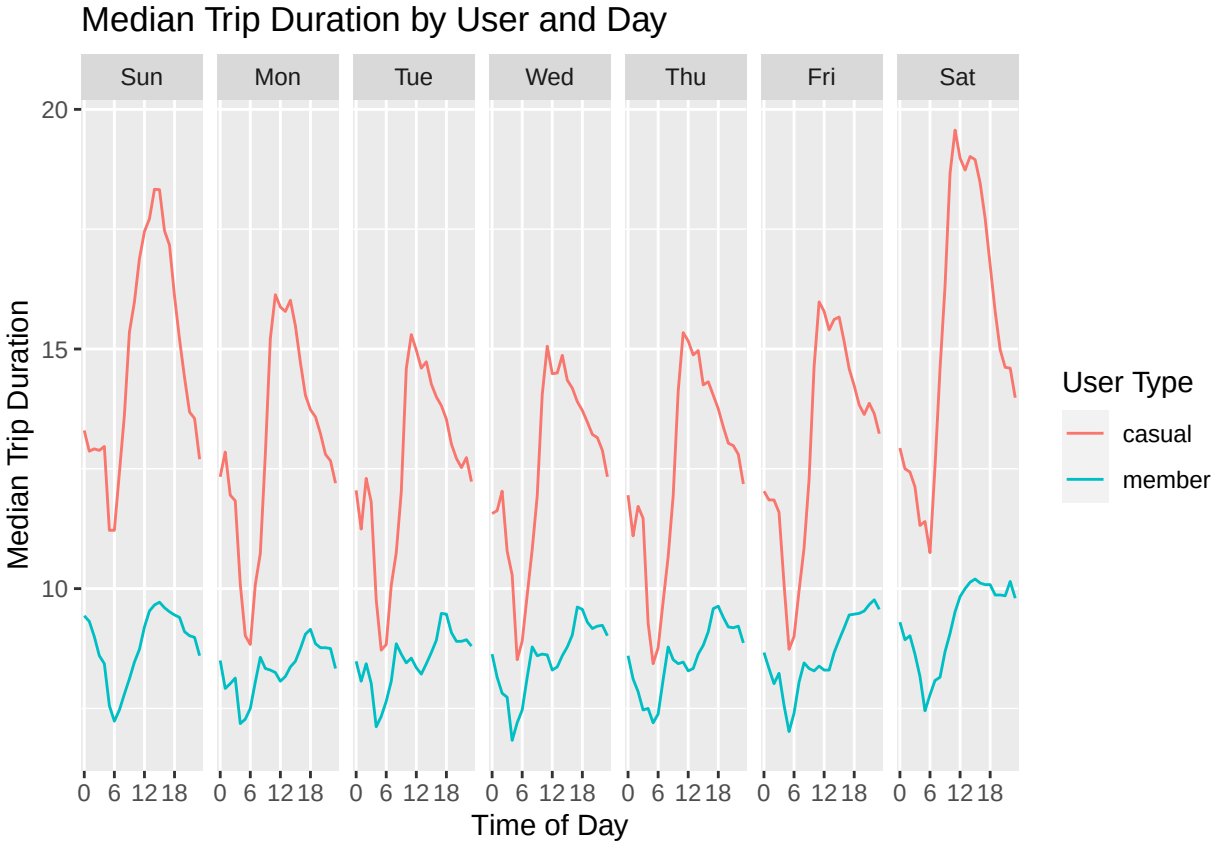
The time-series chart 'Ride Trips by Hour' confirms the higher usage by members with a clear pattern of weekday volumes corresponding to transportation for work purposes. The casual users follow a daily pattern with increases toward the end of the week - Thursday through Saturday. The evening spike of casual users may indicate some casual individuals returning home from work or potentially tourists returning bikes in the early evening before a night out in the city. Poor weather conditions can directly impact the ridership on a given day as exemplified on the days with lower than expected member and casual user counts. A thunderstorm on Sept. 6 and rain throughout early October contributed to the lower bike trip volumes visible in the plot.

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 every week based on member users utilizing the bikes for work and casual users renting the bikes for recreational purposes.

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

Duration of Bike Trips

Next, we evaluate the duration of bike trips to better understand longevity of individual bike unavailability along with reason for the 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. Users will incur an additional fee 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 a peak of greater than 15 minutes on average everyday of the week, particularly around lunch on weekdays.

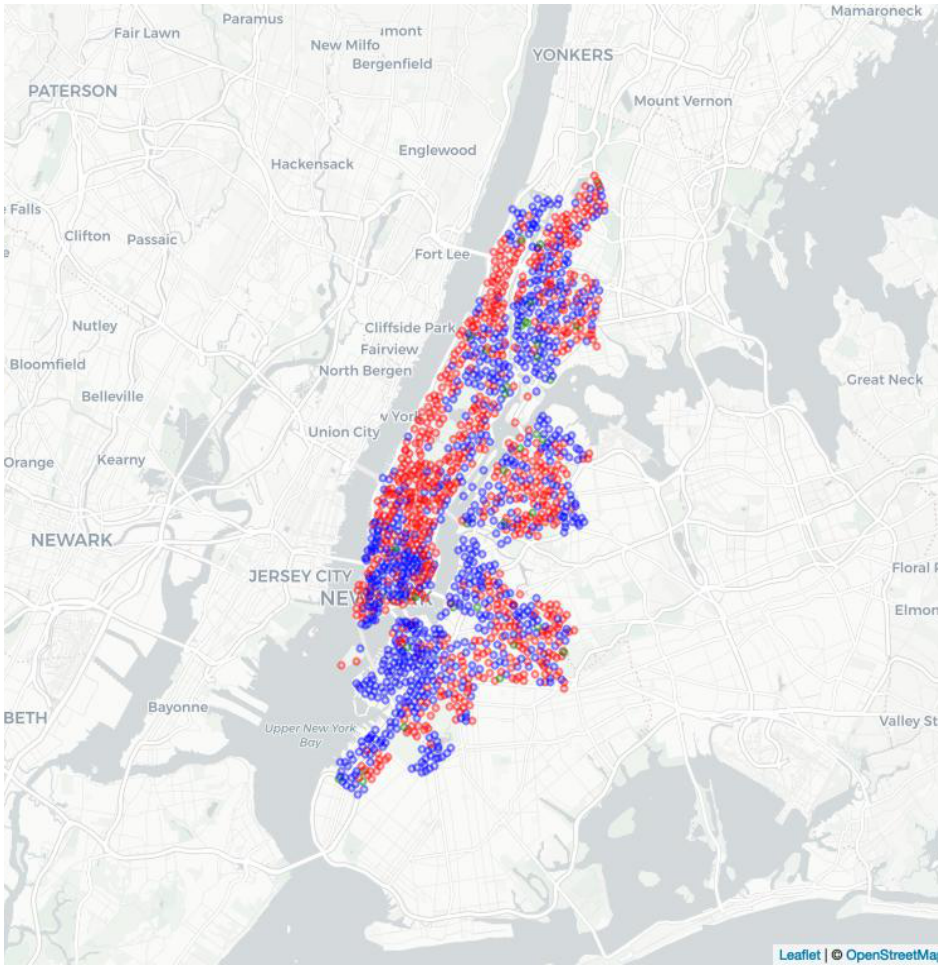
We observe that member users tend to have longer average trips during the morning and evening rush hours on weekdays with a slight deviation on Friday evening as the weekend starts. The afternoon and evening hours of Saturday and Sunday show longer trips for members as likely the result of recreational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

Surplus: Blue −− Shortage: Red −− Even: Green

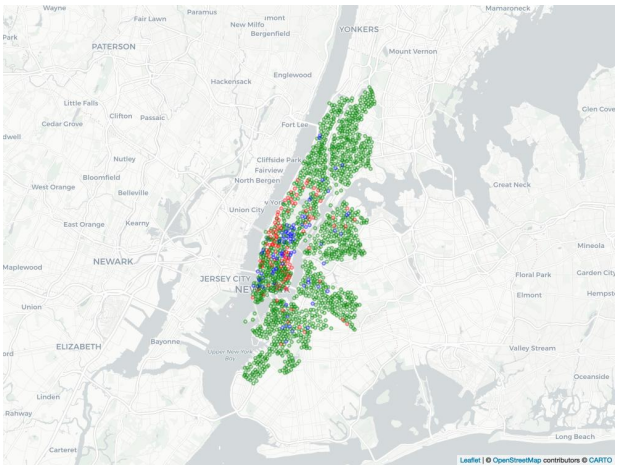


The plot of docking stations across the New York City maps the location of every docking station along with a color to indicate a net positive, net negative or even amount of bikes for the given month. The blue stations are net positive, and red stations indicate net negative while green stations are even for the entire month. Of the 1,716 docking stations represented, 910 ended with a surplus, 771 with a shortage, and 35 with an even count.

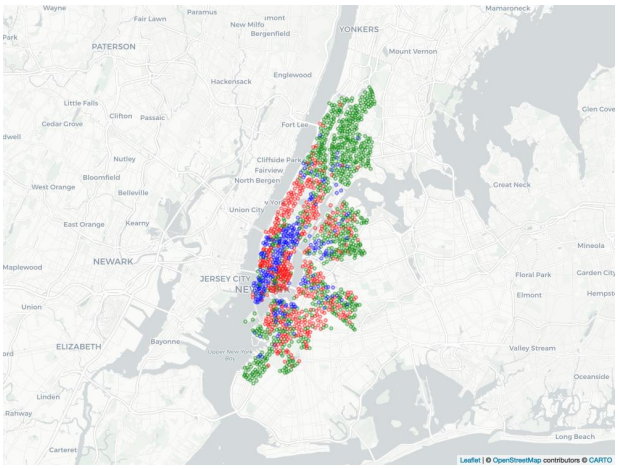
Docking Station Surplus: Weekday Pattern

To better understand the daily flow of bikes, the average arrivals and departures on Wednesdays of the three-month period per docking station are plotted below by hour of the day. The selection of Wednesday, the day of highest use, provides an accurate representation of weekday activity. Starting at 5 a.m., most docking stations are colored green, likely indicating little movement of bikes at such an early hour. At 8 a.m., the docking stations in Midtown and the Financial District, two concentrated are of commercial offices. are primarily blue as individuals make their way to work. Also at 8 a.m., red docking stations populate

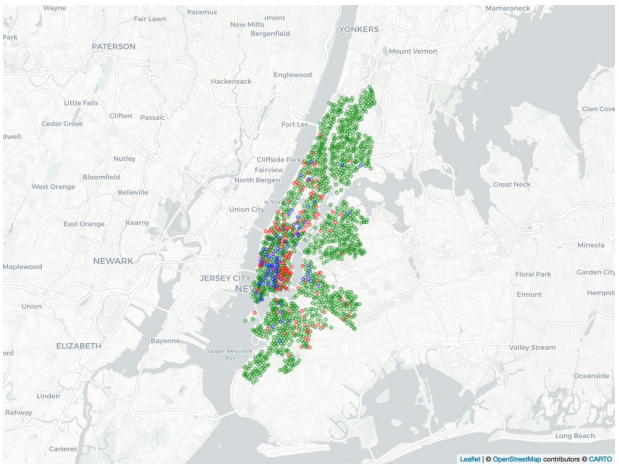
the more residential areas of the upper West Side and the East Village. At 11 a.m., most docking stations are green across the city as the morning rush has subsided. Later at 2 p.m., the plot remains similar to 11 a.m. as the lunch movement has subsided and before evening rush our. Then at 5 p.m., the inverse of the 8 a.m. map appears. Now, Midtown and the Financial District areas are red representing the departures of workers from the commercial centers. Also at 5 p.m., blue docking stations color the Upper West Side and East Village as individuals return home from work. Finally during the 8 p.m. hour, the patterns remains similar to the 5 p.m. plot as bike movement continues away from commercial areas toward residential areas with an increasing number of green docking stations throughout the city.



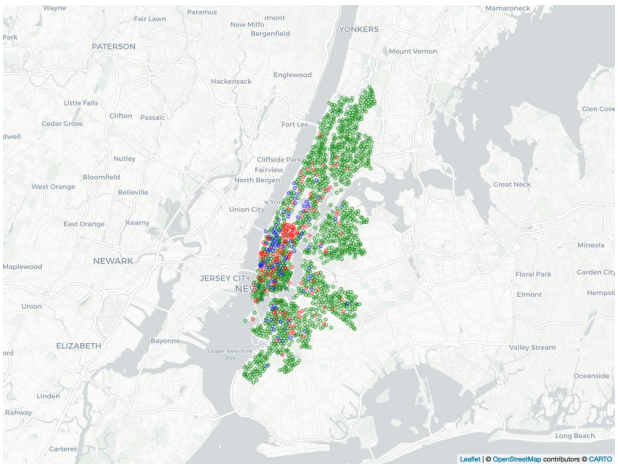
(a) 5 a.m.



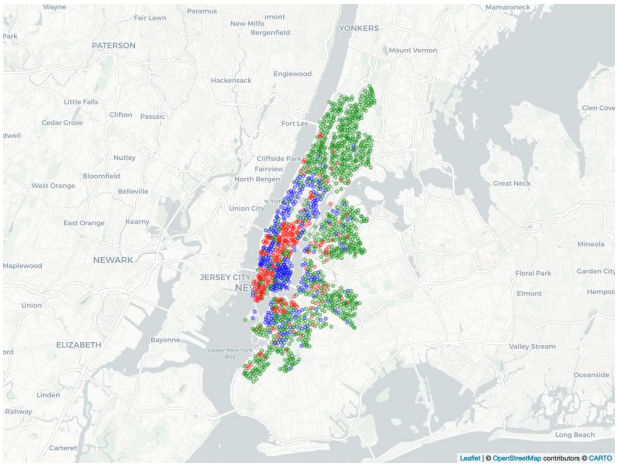
(b) 8 a.m.



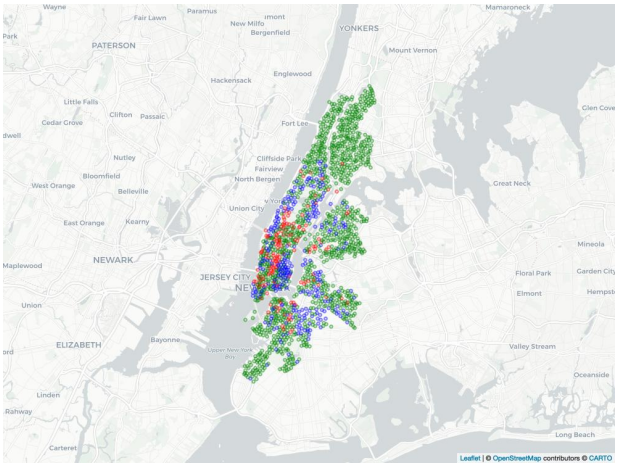
(c) 11 a.m.



(d) 2 p.m.



(e) 5 p.m.

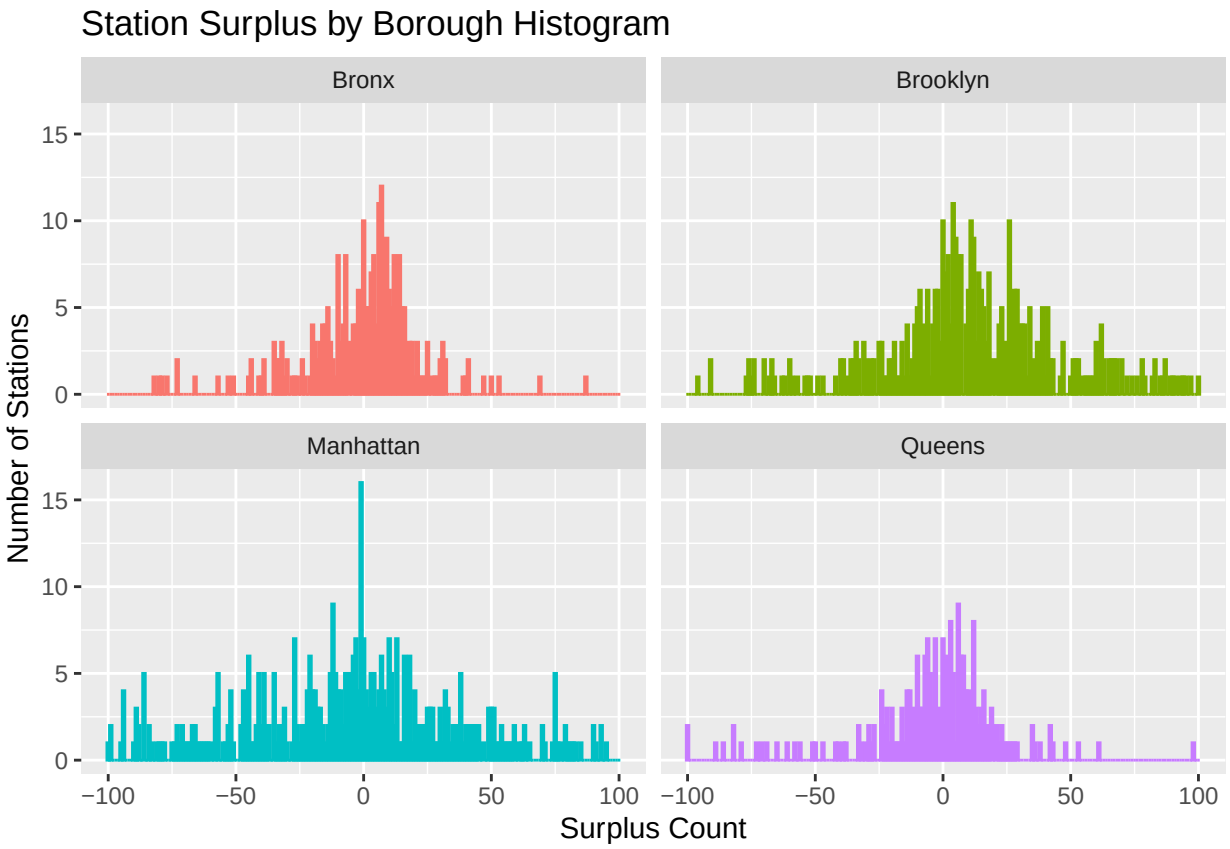


(f) 8 p.m.

Weekday Pattern: 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.

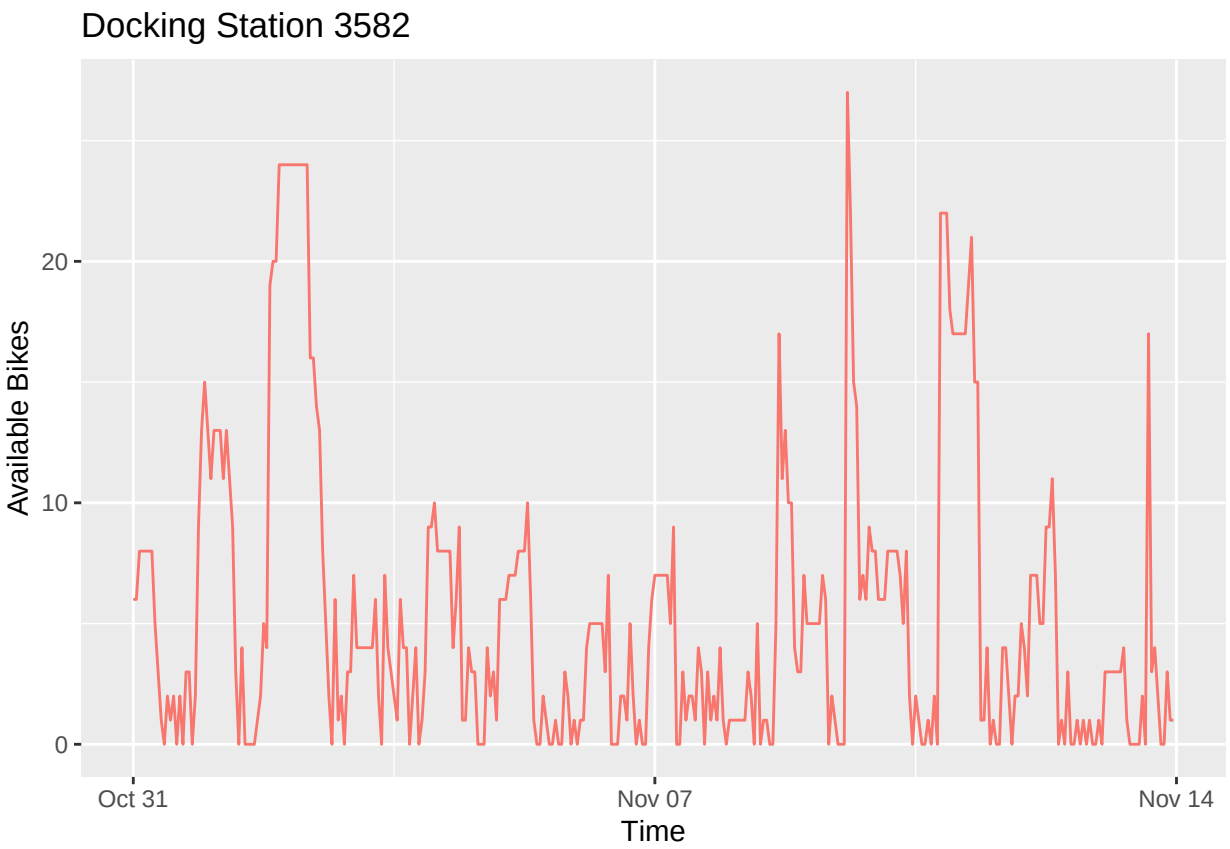


The plot denotes a near normal distribution for the four boroughs included in the dataset — the Bronx, Brooklyn, Manhattan, and Queens. The balanced distribution across the four NYC boroughs likely demonstrates the bike trips are contained within a borough. With average duration less than 10 minutes 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 the bikeshare system.

Rebalancing Example

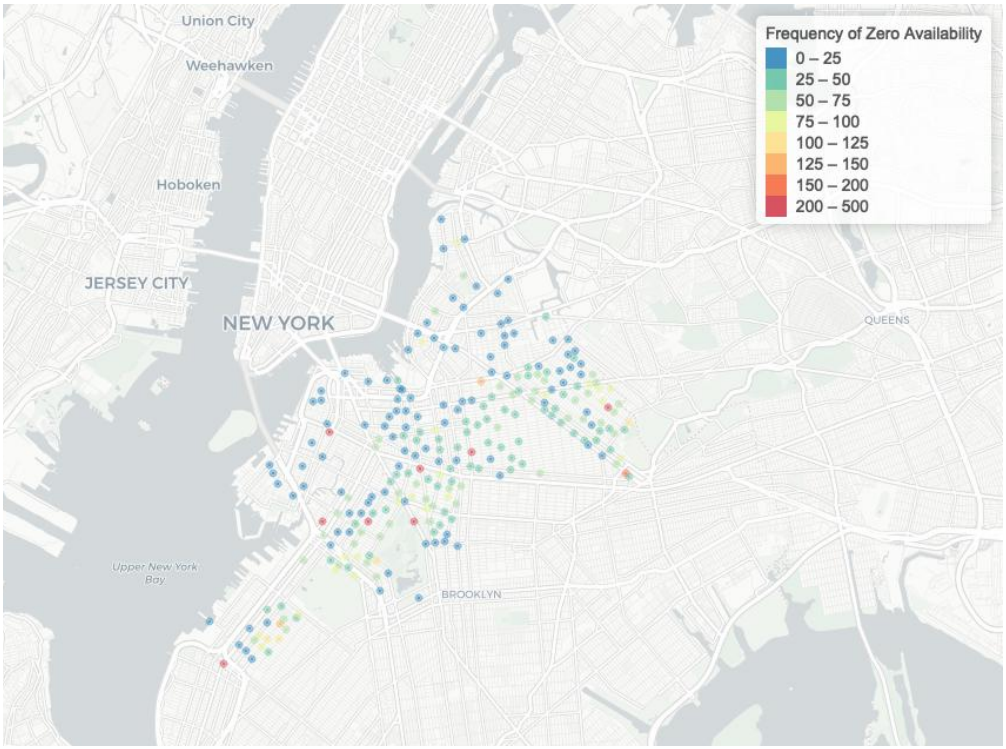
According to the Citi Bike operating report for September 2022 (latest available), the Citi Bike staff rebalanced a total of 70,117 bikes. The operating report also denotes 1,625 active stations at the end of the month. The report also calculates an average bike fleet in September as 24,514.0 with 24,327 bikes in the fleet on the last day of the month. Given the total number of rebalanced bike of 70,117 for the month and an average of 24,514 bikes available per day, then almost 10% of bikes are rebalanced daily.

The line chart depicts the number of available bikes every 1 hour for docking station 3582 in Crown Heights Brooklyn. The chart clearly shows six spikes to 15 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.



Zero Bike Availability

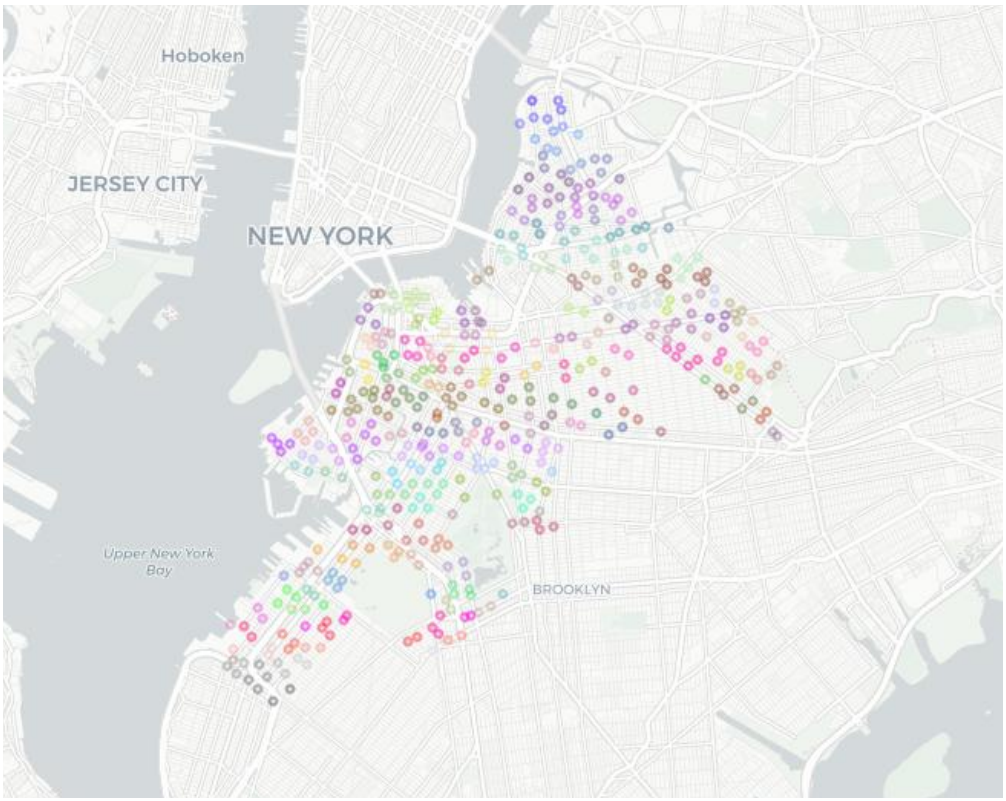
he map plot shows the docking stations with frequency of zero bike availability. Most docking stations have a lower frequency of 50 or less occurrences across the 1,344 15-minute intervals of the two week period. We do notice a number of red stations indicating high frequency of zero available bikes.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



Hierarchical Clustering

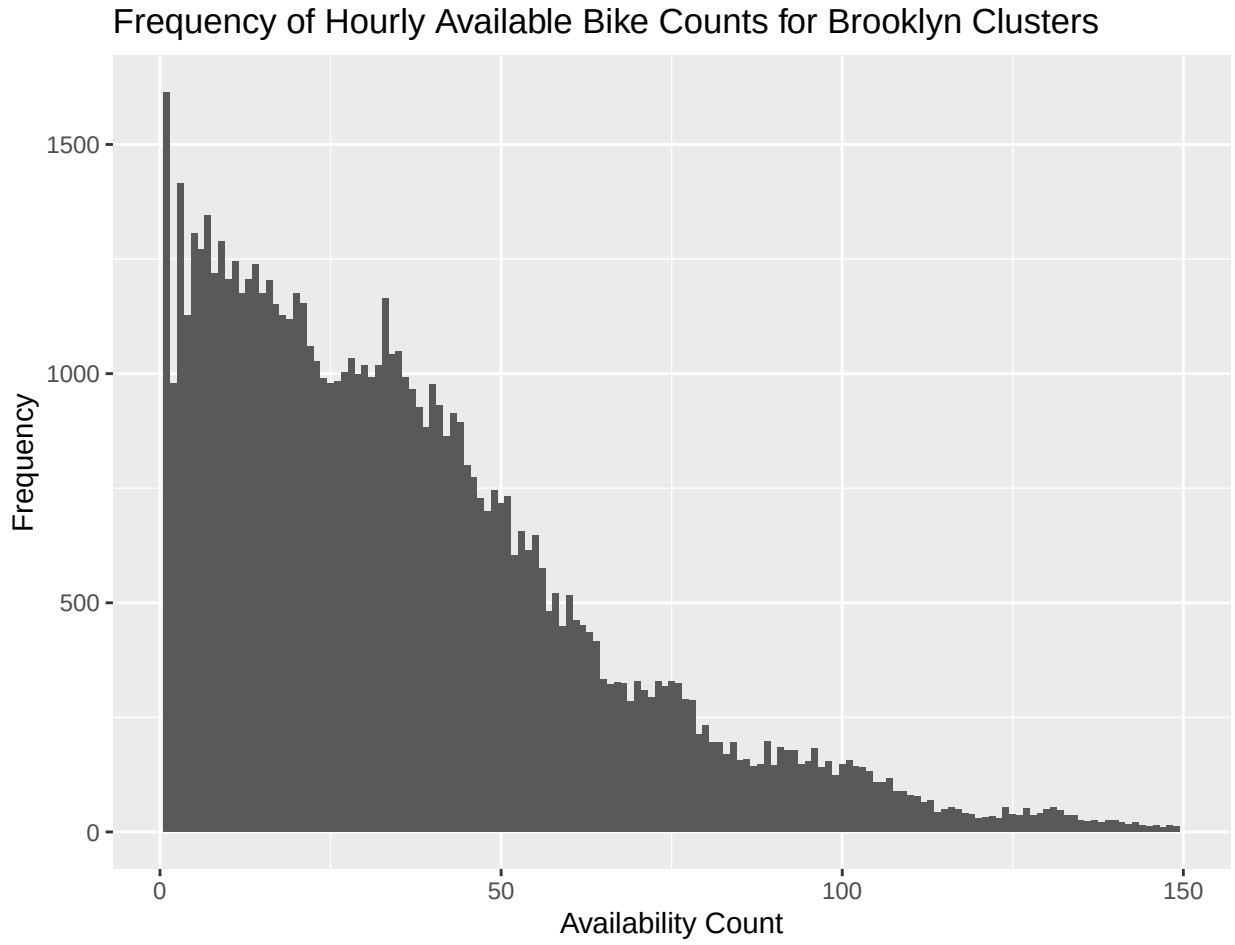
With over 1,700 stations and a walking distance of 400 meters (or one quarter of a mile), the clustering of all NYC docking stations results in over 700 unique clusters. The count of clusters caused vector memory issues in RStudio. In order to continue the research approach, the stations under consideration were decreased to one borough – Brooklyn. Brooklyn contains 474 docking stations. Using the hierarchical clustering from the `stats` R library, with a distance cutoff of 400 meters, the result is 213 clusters or a little over 2 docking stations per cluster. The map plot shows the docking stations colored by cluster number.

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



Bike Availability by Cluster

The histogram shows the frequency of bike availability counts across all the Brooklyn clusters for each 1-hour interval for the two-week timespan. The value of zero is the most frequent value followed by two. A plateau is visible starting around 25 which marks a station capacity. A similar plateau also occurs around 60, close to double the single station capacity which would be expected given most clusters are two docking stations. The plot confirms the likelihood of zero bike availability.



Count Data Regression Modeling

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.

For evaluating the models, the two full weeks of data spanning Oct. 31 through Nov. 13 are partitioned into a training set of 75% and a test set of 25%. 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, and Zero-Inflated.

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.

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

The mean of the bikes available by cluster is 36.34474 and the population variance over the dataset is 908.1704 indicating overdispersion exists.

Generalized Linear Models for Count Data

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. Despite the indication of overdispersion due to the much greater variance than mean, we attempt the Poisson model first.

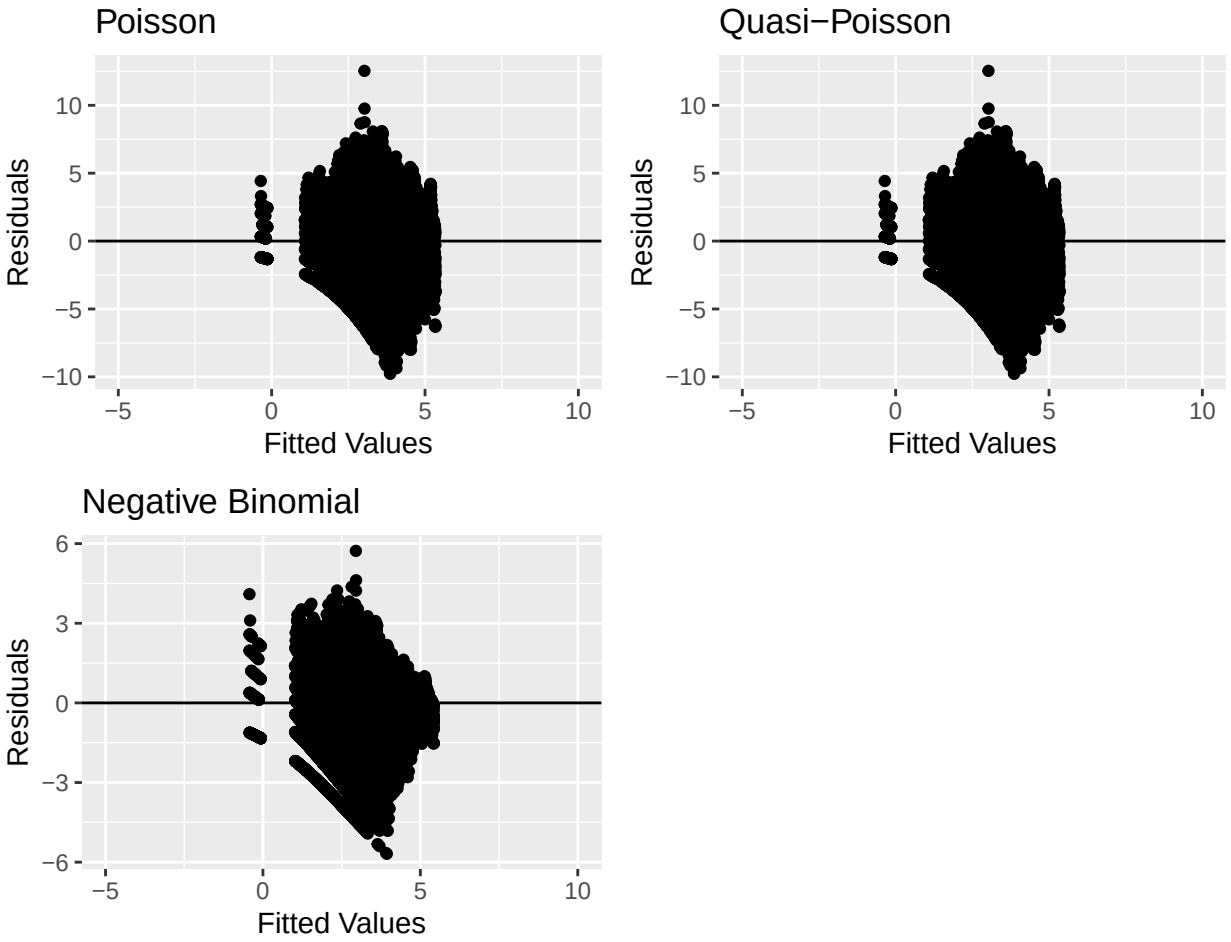
Due to the high number of clusters and hourly intervals used for the regression models, the models create a large volume of dummy variables. For cluster, 212 variables are created, for time, 23 variables created, and for day one additional variable is created. This variable expansion makes the model a bit difficult to assess, as the model only has three input categorical variables.

The `dispersiontest` function from the `AER` library indicates a dispersion of 3.8466699.

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.

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.

Model	Deviance	Degrees.Freedom	AIC
Poisson	219859.7	53616	487809.6
Quasi-Poisson	219859.7	53616	NA
Negative Binomial	66131.4	NA	414441.1



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.

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 11.2482 to 12.0320. These results show that the average distance of approximately 11 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.

Model	RMSE	MAE	MASE	Missed.Zero
Zero (Poisson)	11.2482	8.0648	0.2649	280
Poisson	11.3075	8.1143	0.2665	263
Quasi-Poisson	11.3075	8.1143	0.2665	263
Zero (Neg. Binomial)	11.8260	8.3998	0.2759	280
Negative Binomial	12.0320	8.5401	0.2805	263

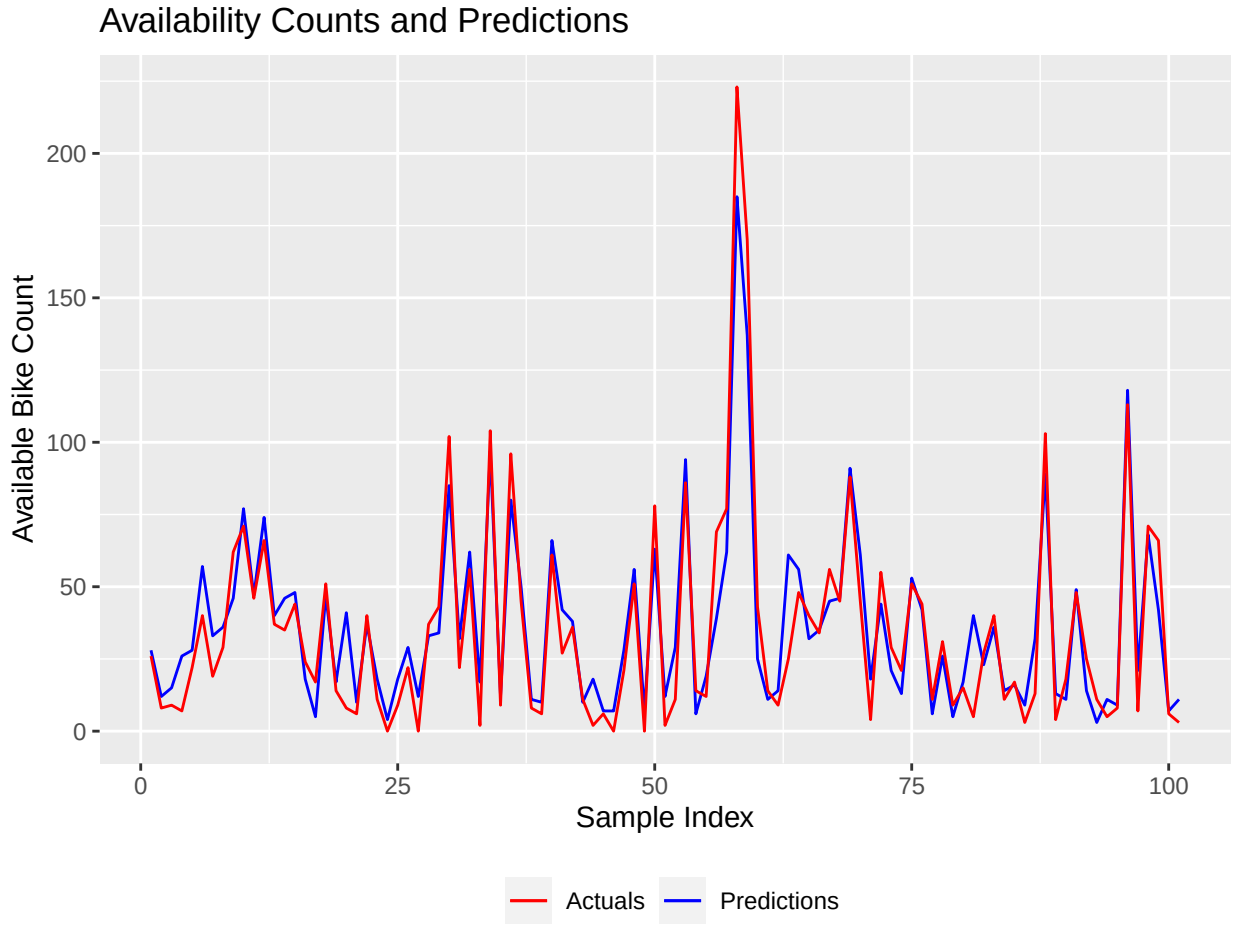
In the end we selected the Quasi-Poisson model because it resulted in the second best RMSE and MAE and had the fewest missed Zero counts. The Quasi-Poisson regression equation is presented for the selected model.

Selection: Quasi-Poisson regression equation

$$log(bikes) = 3.9063 + \beta_1 \cdot cluster + \beta_2 \cdot time + \beta_3 \cdot day \tag{1}$$

Model Results Visualization

The test data dataset consisted of over 17,000 rows for prediction. In order to better understand the predictions from the Poisson model against the actual test data values, the plot displays a subset of 100 predictions and the corresponding actual values. The actual values in red extend to the 0 axis while the blue line does not reach the 0 axis as often, again indicating the inability of the model to accurately predict the true zero values. Upon visual inspection, the lines for prediction and actual values do appear consistent.



Note on other model considerations: For the predictors, we finalized the inputs as longitude and latitude, anticipating the end user would actually be providing 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 again, due to system constraints the time interval of one hour was selected. 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.

Also considered, but not used was the time series forecasting model approach. Because the forecasting did not allow for instance predictions and would have relied on consistent up-to-date info, we decided not to include the forecasting approach in the final report.

Real-World Prediction Proof of Concept

With an eye toward making the model real-world functional, a simple `R` function was written to predict the number of bikes available based on the user-friendly inputs. First, the longitude and latitude are translated into a cluster identifier based on a nearest neighbor search of the docking stations. Then, with the cluster identifier and other user inputs, the GLM regression model is executed to predict the number of available bikes. The result 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 user-friendly inputs:

- Latitude and longitude (location on map)
- Weekday or Weekend
- Hour of the day

Neighborood	Longitude	Latitude	Time	Day	Prediction	Cluster
Crown Heights	-73.9609	40.6736	22:00:00	WDAY	13	58
Williamsburg	-73.9617	40.7192	18:00:00	WKND	57	201
Sunset Park	-74.0051	40.6434	17:00:00	WDAY	5	10

Conclusion & Next Steps

Using the data provided by the Citi Bike operating system, we constructed a model to predict Citi Bike availability for a given day and time in Brooklyn with the possibility of extension across all of New York City. Using data science principles and common model algorithms, a valuable prediction model was constructed for a user-friendly interaction. Based on location, day of the week, and time of day, the combination of a hierarchical clustering and Poisson count model algorithm can predict the number of available Citi Bikes within a quarter mile. The overall prediction model provides potential real-world application for individuals planning transportation options ahead of time across New York City.

In assessing the count models, the Poisson model was selected based on its RMSE value and ability to accurately predict zero availability. The false positive of bike availability when zero bikes available would greatly impact user experience and thus played a key factor in the model selection decision.

The exploratory analysis of the Citi Bike dataset for NYC-based trips in August through October 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.

Future Work

Given the working prediction model, the future consideration is the deployment of the prediction model as a simple website or smartphone app to get the model into users' hands for real-world feedback. With the use of the model on a smartphone, the location input could come directly from the GPS location of the device to ensure a truly user-friendly experience.

Due to the system constraints and resulting model decisions, we believe several opportunities to improve the model accuracy can be followed. First, we believe a higher accuracy of zero availability predictions would provide a better user experience. We recognize a prediction of 10 available bikes instead of 8 would not be the headache to a user, whereas a prediction of 2 available bikes instead of 0 would be quite frustrating. Our first direction of improvement would be the improvement of accurate zero bike predictions.

To give the model and greater user base, the obvious next step would be the inclusion of a model covering all the docking stations of New York City. With the majority of docking stations in Manhattan, a model of the entire city would be provide value to a greater audience. Currently, the clustering of docking stations is static, so to improve bike availability based on location, a real-time clustering approach to identify all docking stations within a quarter mile of the user at the time of use would be beneficial. Finally, the researchers understand the limitation exerted by the two-week availability data, so to improve the model, a greater timeframe of availability data we anticipate would improve the model.

Auto-regressive Spatio-XXX

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

References

Hosseinzadeh, Aryan, Abolfazl Karimpour, and Robert Kluger. 2021. "Factors Influencing Shared Micromobility Services: An Analysis of e-Scooters and Bikeshare." *Transportation Research Part D: Transport and Environment* 100: 103047. https://doi.org/10.1016/j.trd.2021.103047.

Hyland, Michael, Zihan Hong, Helen Karla Pinto, and Ying Chen. 2018. "Hybrid Cluster-Regression Approach to Model Bikeshare Station Usage." *Transportation Research Part A: Policy and Practice* 115: 71–89. https://doi.org/10.1016/j.tra.2017.11.009.

Li, Aoyong, Yizhe Huang, and Kay W. Axhausen. 2020. "An Approach to Imputing Destination Activities for Inclusion in Measures of Bicycle Accessibility." *Journal of Transport Geography* 82: 102566. https://doi.org/10.1016/j.jtrangeo.2019.102566.

Lin, Lei, Zhengbing He, and Srinivas Peeta. 2018. "Predicting Station-Level Hourly Demand in a Large-Scale Bike-Sharing Network: A Graph Convolutional Neural Network Approach." *Transportation Research Part C: Emerging Technologies* 97: 258–76. https://doi.org/10.1016/j.trc.2018.10.011.

Ma, Xinwei, Yanjie Ji, Yuchuan Jin, Jianbiao Wang, and Mingjia He. 2018. "Modeling the Factors Influencing the Activity Spaces of Bikeshare Around Metro Stations: A Spatial Regression Model." *Sustainability* 10 (11): 3949. https://doi.org/10.3390/su10113949.

Médard de Chardon, Cyrille, Geoffrey Caruso, and Isabelle Thomas. 2016. "Bike-Share Rebalancing Strategies, Patterns, and Purpose." *Journal of Transport Geography* 55: 22–39. https://doi.org/10.1016/j.jtrangeo.2016.07.003.

Noland, Robert B., Michael J. Smart, and Ziye Guo. 2016. "Bikeshare Trip Generation in New York City." *Transportation Research Part A: Policy and Practice* 94: 164–81. https://doi.org/10.1016/j.tra.2016.08.030.

Qian, Xiaodong, Miguel Jaller, and Giovanni Circella. 2022. "Equitable Distribution of Bikeshare Stations: An Optimization Approach." *Journal of Transport Geography* 98: 103174. https://doi.org/10.1016/j.jtrangeo.2021.103174.

Wang, Kailai, and Yu-Jen Chen. 2020. "Joint Analysis of the Impacts of Built Environment on Bikeshare Station Capacity and Trip Attractions." *Journal of Transport Geography* 82: 102603. https://doi.org/10.1016/j.jtrangeo.2019.102603.

Wang, Xize, Greg Lindsey, Jessica E. Schoner, and Andrew Harrison. 2016. "Modeling Bike Share Station Activity: Effects of Nearby Businesses and Jobs on Trips to and from Stations." *Journal of Urban Planning and Development* 142 (1). https://doi.org/10.1061/(asce)up.1943-5444.0000273.

Zhou, Xiaolu, Mingshu Wang, and Dongying Li. 2019. "Bike-Sharing or Taxi? Modeling the Choices of Travel Mode in Chicago Using Machine Learning." *Journal of Transport Geography* 79: 102479. https://doi.org/10.1016/j.jtrangeo.2019.102479.

Appendix with Code

```
knitr::knit_hooks$set(time_it = local({
  now <- NULL
  function(before, options) {
    if (before) {
      # record the current time before each chunk
      now <<- Sys.time()
    } else {
      # calculate the time difference after a chunk
      res <- difftime(Sys.time(), now)
      # return a character string to show the time
      paste("Time for this code chunk to run:", res)
    }
  }
}))

#knitr::opts_chunk$set(time_it = TRUE)

# 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)
library(cowplot)

# 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='Median Number of Bike Trips') +
  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",
       x = "Time by Hour",
       y = "Ride Trips Counts") +
  guides(colour = guide_legend(title = "User Type") ) +
  theme(legend.position="bottom")
citibike_by_hour_ts %>%
  gg_season(cnt, period = "week") +
  labs(x = "Time by Hour",
       y = "Bike Trips",
       title = "Seasonal plot: Weekly Trip Counts") +
  theme(legend.position="none")
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",
       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)))
```

```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))

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
par(mar = c(0.1, 4.1, 1.1, 4.1))
img <- readJPEG("stations_surplus_sum.jpg")
plot(1:10,ty="n", axes = 0, xlab='', ylab='', asp=1.0)
p_title <- 'Overall Surplus/Shortage by Docking Station'
p_subtitle <- "Surplus: Blue -- Shortage: Red -- Even: Green"
title(side = 3, line=-1, adj=0, p_title)
mtext(side = 3, line=-3, adj=0, p_subtitle)
rasterImage(img,-1,0,12,10)
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))

stations_attrs_trim_sur %>%
  filter(surplus.sum >= -100 & surplus.sum <= 100 & boro_name %in% c('Bronx', 'Brooklyn', 'Manhattan',
  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",
       fill = "Borough",
       color='Borough') +
  theme(legend.position="none")
# 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 & endsWith(as.character(timestamp), "00:00"))
#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")

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)

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")
```

```r
# 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)

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_

# 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(t

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
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
```

```r
# POISSON
mod_1H_pois <- glm(bikes.avail.count ~ cluster + time + day, data = train_1H, family ="poisson")
mod_1H_pois_sum <- summary(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))
mase_mod_1H_pois <- mase(test_1H$bikes.avail.count, floor(pred_1H_pois))
mod_1H_pois_dev <- mod_1H_pois_sum$deviance
mod_1H_pois_df <- mod_1H_pois_sum$df[2]
mod_1H_pois_aic <- mod_1H_pois_sum$aic
p_resid_pois <- ggplot(mod_1H_pois, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  labs(title='Poisson', x='Fitted Values', y='Residuals')

p_resid_pois <- p_resid_pois +  xlim(-5, 10)
disp_test <- dispersiontest(mod_1H_pois)
# This indicates dispersion
disp_val <- as.numeric(disp_test$estimate[1])
# QUASI-POISSON
mod_1H_qp <- glm(bikes.avail.count ~ cluster + time + day, data = train_1H, family ="quasipoisson")
mod_1H_qp_sum <- 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))
mase_mod_1H_qp <- mase(test_1H$bikes.avail.count, floor(pred_1H_qp))
p_resid_qp <- ggplot(mod_1H_qp, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  labs(title='Quasi-Poisson', x='Fitted Values', y='Residuals')

p_resid_qp <- p_resid_qp +  xlim(-5, 10)
mod_1H_qp_dev <- mod_1H_qp_sum$deviance
mod_1H_qp_df <- mod_1H_qp_sum$df[2]
mod_1H_qp_aic <- mod_1H_qp_sum$aic
# NEGATIVE BINOMIAL
mod_1H_nb <- glm.nb(bikes.avail.count ~ cluster + time + day, data=train_1H)
mod_1H_nb_sum <- 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))
mase_mod_1H_nb <- mase(test_1H$bikes.avail.count, floor(pred_1H_nb))
mod_1H_nb_dev <- mod_1H_nb$deviance
#mod_1H_nb_df <- mod_1H_nb$df[2]
mod_1H_nb_aic <- mod_1H_nb$aic
p_resid_nb <- ggplot(mod_1H_nb, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0) +
  labs(title='Negative Binomial', x='Fitted Values', y='Residuals')
```

```
p_resid_nb <- p_resid_nb +  xlim(-5, 10)
deviances <- c(mod_1H_pois_dev, mod_1H_qp_dev, mod_1H_nb_dev)
degfrees <- c(mod_1H_pois_df, mod_1H_qp_df, "NA")
aics <- c(mod_1H_pois_aic, mod_1H_qp_aic, mod_1H_nb_aic)

models_stats <- c("Poisson","Quasi-Poisson","Negative Binomial")

data.frame(Model=models_stats, Deviance=deviances, Degrees.Freedom=degfrees, AIC=aics) %>%
  kable(digits = 1) %>%
  kable_paper("hover", full_width = F) %>%
  kable_styling(latex_options = "HOLD_position")
plot_grid(p_resid_pois, p_resid_qp, p_resid_nb)
# ZERO INFLATED: POISSON
mod_1H_zero_pois <- zeroinfl(bikes.avail.count ~ cluster + time + day, data=train_1H, dist = "poisson")
mod_1H_zerop_sum <- 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))
mase_mod_1H_zero_pois <- mase(test_1H$bikes.avail.count, floor(pred_1H_zero_pois))
# ZERO INFLATED: NEGATIVE BINOMIAL
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))
mase_mod_1H_zero_nb <- mase(test_1H$bikes.avail.count, floor(pred_1H_zero_nb))
rmse_1H <- c(rmse_mod_1H_pois, rmse_mod_1H_qp, rmse_mod_1H_nb,
             rmse_mod_1H_zero_pois, rmse_mod_1H_zero_nb)

# Calculate normalized RMSE with 237 as the max bike avail count for test data
rmse_1H_norm <- rmse_1H/237

mae_1H <- c(mae_mod_1H_pois, mae_mod_1H_qp, mae_mod_1H_nb,
            mae_mod_1H_zero_pois, mae_mod_1H_zero_nb)

mase_1H <- c(mase_mod_1H_pois, mase_mod_1H_qp, mase_mod_1H_nb,
             mase_mod_1H_zero_pois, mase_mod_1H_zero_nb)

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

pred_zero_1H <- c("163","163","163","133","133")

models_1H <- c("Poisson","Quasi-Poisson","Negative Binomial",
               "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)
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])
data.frame(Model=models_1H, RMSE=rmse_1H, MAE=mae_1H, MASE=mase_1H, Missed.Zero=miss_zero_1H) %>%
  arrange(rmse_1H) %>% kable(digits = 4) %>%
  kable_paper("hover", full_width = F) %>%
  kable_styling(latex_options = "HOLD_position")
# 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="Predi
  geom_line(data = df_for_plot_trim, aes(x = 1:nrow(df_for_plot_trim), y = Actual, color = "Actuals")) +
  labs(x='Sample Index',
       y='Available Bike Count',
       title='Availability Counts and Predictions') +
  scale_color_manual(values = c("Predictions" = "blue", "Actuals" = "red")) +
  theme(legend.position="bottom",
        legend.title=element_blank())

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])
hoods <- c("Crown Heights", "Williamsburg", "Sunset Park")
lons <- c(-73.960859, -73.9617, -74.00509)
lats <- c(40.67355, 40.7192, 40.64338)
times <- c("22:00:00", "18:00:00", "17:00:00")
days <- c("WDAY", "WKND", "WDAY")
preds <- c(prediction_CH[1], prediction02_WB[1], prediction02_SP[1])
clusts <- c(prediction_CH[2], prediction02_WB[2], prediction02_SP[2])


data.frame(Neighborood=hoods, Longitude=lons, Latitude=lats, Time=times, Day=days, Prediction=preds, Clu
  kable_paper("hover", full_width = F) %>%
  kable_styling(latex_options = "HOLD_position")
# https://yihui.org/en/2018/09/code-appendix/
```