Optimizing CitiBike Docking Strategy Utilizing Modeling and Simulation

Eddieb Sadat
5-16-2022
Stevens Institute of Technology

Dr. Paul Grogan
SYS-611

I pledge my honor that I have abided by the Stevens Honor System
E.S.

Table of Contents

1. Introduction

In a world where consumers demand the most convenient and fast services, it is increasingly important for companies to constantly improve their services to stay afloat in the market. In the case of CitiBike, customers are able to utilize their services by renting a bike to quickly and conveniently travel across New York City, where driving is often a nightmare. Bike stations are spread across the city, and customers are able to rent a bike from one station and return it at any other station. Ideally, CitiBike should build very large stations to have enough bikes for every renter and to have enough empty docks for all bike returns. However, this is an extremely inefficient method for CitiBike, since they will have to produce and maintain unused docks and bikes, adding to avoidable costs (not to mention that paying for large real estate in NYC is extremely expensive). The best option for both CitiBike and their customers, is to instead employ the use of modeling and simulation to develop an optimized system that will maximize customer utility while minimizing CitiBike's expenses. This paper attempts to create an optimized system and documents the system objectives, the implementation of models and simulation to accomplish the objectives, and the results and takeaways throughout the process.

1.1 Study Objectives

There are many ways to interpret what gives customers utility and what increases costs for CitiBike. For this project, I determined that customers will achieve maximum satisfaction when they are able to rent a bike out immediately and when they are able to return a bike immediately. In other words, there must never be an undersupply of bikes (where there is a customer arrival but no bikes left at the station) and there must never be an overflow of bikes (where there is a customer return but no free docks left at the station). Undersupply can be managed by having enough docks at a station (the more docks, the more potential bikes available), and overflow can be managed by leaving enough empty docks at the beginning of the day to allow for more potential customer returns. Thus, the objective of this project is to minimize undersupply and overflow by determining the number of docks needed at the station, and how many docks to fill up with bikes at the beginning of each day.

1.2 System Boundaries

Keeping in mind that this project has a rushed deadline, I have made several assumptions about the system, many of which are to simplify the complexity of the real-life system. First, I am going to only be looking at one station with data from only the weekends (Saturday and Sunday) of April from 12-4pm. Using this data, a generator will be created to determine the customer arrivals (renting a bike) and customer returns (returning a bike). Thus, throughout the model, the number of bikes at a station will change, and subsequently, the number of full/empty docks. Besides customer interaction, no other factor will affect the number of bikes/docks available; it is assumed that the station is filled/emptied to a specific number of bikes only at the start of the day (on initialization). The number of docks will not change throughout the system simulation, and will only change per model as the initialized value is altered. Lastly, it is assumed that service time is instantaneous (assuming there is a bike or empty dock available for rent or return) since the customers simply scan to rent a bike or push the bike into the dock. So long as the customer is able to complete the task they want (rent or return), the service time is negligible to the scope of the system objectives.

1.3 Key Performance Measures

The satisfaction of both the customers and the CitiBike organization is what will measure the success of the system. As stated above, customers will be most happy when they are able to rent and return a bike from the station. Therefore, minimizing the undersupply and overflow are two key performance measures.

2. Modeling Approach

2.1 Data Collection & Processing

One reason I opted to complete this specific project was because CitiBike compiles all their bike and station data, and makes it readily available in an open repository [1]. Looking through the repository, I decided to choose the most recent data available at the time, which was the full aggregate data from April 2022. Downloading the file saves a .csv with all the collected data. Since I am using Python, I used the PANDAS library to save the file into a dataframe, which will allow me to view and edit all the data very easily. Displaying the data frame, I found that there were over 2.3 million data entries, with each entry containing the following information[1]:

- ride_id → Ride ID
- rideable_type → Type of Bike (electric or classic)
- started_at → Date & Time when bike is removed from a station
- ended_at → Date & Time when bike is returned to a station
- start_station_name → Name of station where bike is removed
- start_station_id → ID of station where bike is removed
- end_station_name → Name of station where bike is returned
- end_station_id → ID os station where bike is removed
- start_lat → Latitude of station where bike is removed
- end_lat → Latitude of station where bike is returned
- start_lng → Longitude of station where bike is removed
- end_lng → Longitude of station where bike is returned
- member_casual → Customer Type (member or casual)

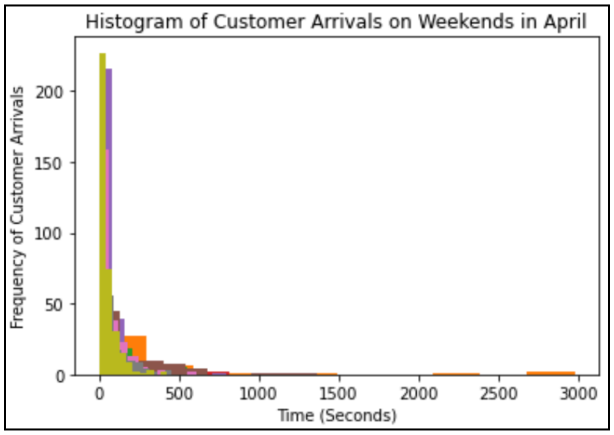
For the scope of this system, the only data that is required are the started_at, ended_at, start_station_name, and end_station_name, and thus all other data columns were dropped[2]. Additionally, the remaining data was filtered to delete any rows with NAN values. To crudely test the integrity of the data, I ran a test to make sure that the number of entries for 'start_station_name' were equal to the 'end_station_name' (to make sure the number of bikes rented is equal to the number of bikes returned), which they were.

Next, I had to choose which station to focus on. I knew I wanted to select one located in Central Park, so I filtered all the data entries in 'started_at' that contained "Central Park" and selected the one with the most data available, which happened to be the station located on Central Park S & 6 Ave[3].

The final step in the data processing was to collect all the data from 12-4pm on the weekends (Saturdays and Sundays). Using google calendar, I made note of which dates correspond to the weekends. I then ran a filter to save all the rows where the start_station_name was Central Park S & 6 Ave into a new data frame. Finally, using multiple filters and the DateTime library [2], I was able to produce a list of all the times between each customer arrival (renting a bike from the station) for each weekend day. Using the same methodology, I also compiled a list of all the times between each customer return (returning a bike to the station) for each weekend day.

2.2 Formulating & Developing the Model

With the real-life data of the interarrival times of customers renting and returning bikes, I created a histogram of each to identify which type of distribution most likely fits the model (Fig 1 & 2).



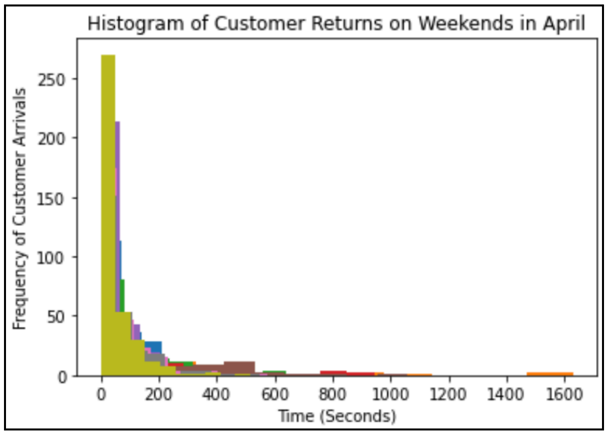
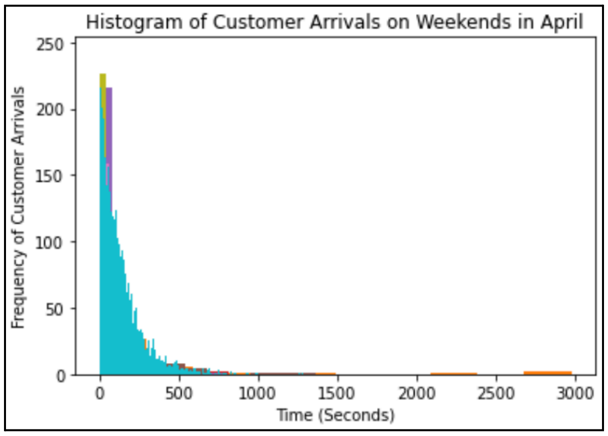


Fig 1 & 2 - Histograms of Customer Arrivals on Weekends in April

I knew visually that they seemed to be an exponential distribution - which confirmed my thoughts that these are arrivals of customers and would likely be a Poisson (exponential) distribution. The next step was to identify which lambda (rate) would create the best fitting model. I note that I tried to utilize the log likelihood method to find the optimized values for the best-fitting model, but I could not seem to get it to work with an exponential distribution.

Instead, I decided to do the brute-force method of creating an IVT generator and overlaying the resulting histogram on top of the ones in Fig 1 & 2, and visually determine the best fitting models (Fig 3 & 4). Using this method, I found that I produced the best model for customer arrivals using a uniform random number generator, a lambda of 0.008, and a multiplication transform of 1.1 (multiplying the resulting generated time by 1.1). Through this same method, I found that using a uniform random number generator, a lambda of 0.015, and a power transform of 1.05 (taking the resulting generated time to the power of 1.05).



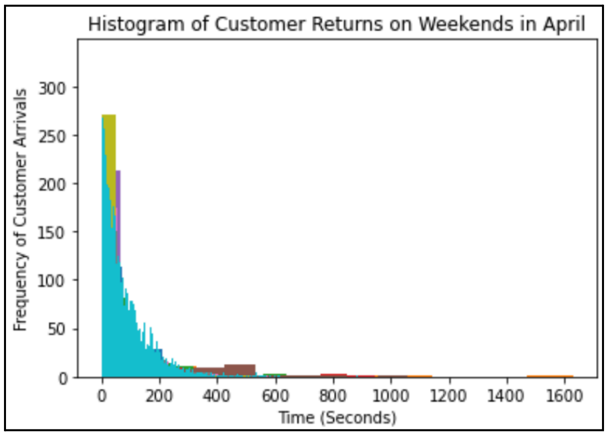


Fig 3 & 4 - Histograms of Customer Arrivals on Weekends in April with IVT Overlay

Unfortunately, after this point I was no longer able to produce a properly working model. Due to the time constraints (so many finals!) I faced while already on a relatively tight deadline, I was unable to look through enough documentation to fully understand how the Simpy library works; it has formatting that I am very unfamiliar with. Sadly I was only able to fully immerse myself into this project a week into May, and by the time I had reached this part of Simpy it was also too late to ask for help. However, I will describe what I had in mind to create, and include details of where it exactly fell apart.

I knew that I needed to create a discrete event simulation, where the timespan of the simulation would run for 4 hours (4*60*60 to convert to seconds for consistency) as to replicate the same timespan from the collected data. First, I needed two decision variables: the number of docks and the number of initial bikes (minimum of zero and maximum equal to number of docks), where the docks would be a resource since they are limited and are exclusively unavailable if there is a bike parked in it, and where the number of bikes would be a container since they can be added and removed from the environment (similar to the spares in the inventory problem from SYS-611 lecture). Second, I would need a function to determine the transitions in number of bikes at the station, which would decrease by one for each customer arrival and increase by one for each customer return. The customer arrival times and return times would be generated using the same generators that were developed above in Fig 3 & 4. Third, I needed to create a function to handle negging, wherein if a customer arrived and no bikes were available or if a customer returned and no docks were free, the customer will not wait and proceed to immediately leave the station. Accompanied will be counters for undersupply (arrival with no bikes available) and overflow (return with no docks available). These counters will be the key performance measures, where the less both counters are the better the model.

2.3 Model Validation & Documentation

Unfortunately as stated above, I was unable to create a working model, and subsequently, am unable to validate it. However, I will describe the process I will have gone by to validate the model and how I would properly document the findings.

Because we have actual data from the CitiBike station, we can verify that the arrival and return time generators are an accurate representation, and thus, can validate that the results are accurate to what the real-life system would produce if we could easily change the number of docks and initial number of bikes. However, solely based on the scope of the simulation that would have been created, it is likely that this model would not be incredibly useful for CitiBike to use as a tool to implement change to their stations. There are simply too many factors that are unconsidered in this model that could drastically change the results of the simulation. For example, the current model only takes into consideration that docks and number of initial bikes can be changed. However, in real life, the location of the station can change (which will result in varying customer arrival and return rates), there can be refilling of bikes throughout specific hours of the day as opposed to only refilling at the start of the day, some customers may decide to wait for a bike or dock instead of leaving, the model only considers a four-hour time span on the weekends while neglecting all other hours and the weekdays (which will drastically change the arrival and return rates), the fact that there are normal and electric bikes which will incur different costs for docking (recharging costs money) and will have different demands, and a slew of many more factors far too long to list here. As a result, this model is likely not very useful besides for speculatory conclusions.

Graphs of the undersupply and overflow can be created to visually see the cumulative values over time. The data provided by CitiBike does not state how many bikes are remaining at each dock, nor does it show the number of customers who missed out on renting or returning a bike. To truly get a good comparison, I would have to gather my own field-data by going to the specific station and manually recording the number of undersupply, overflow, and number of bikes at any given point in time. Using this field-data, I can compare the model graph to the field-data graph and document the comparisons and conclusions. This comparison can also help determine the validity of the model by determining how realistic it is to the real-life system.

3. Results & Analysis

This section is quite difficult to discuss without having a working model to produce results for analysis. However, I will attempt to describe the methodology of how I would have completed this given a functional model.

3.1 Study Design & Conditions

Because I have two decision variables (number of docks and number of initial bikes), I would create a two-dimensional grid with values for both variables ranging from zero to fifty (fifty seems like a lot, but it is also small enough that it is likely reasonable for CitiBike to implement, especially considering how spacious Central Park is). Then, by trying different variations of both variables across the grid, I can generate values for each point on the grid. Theoretically, there will be a point on the grid that contains the smallest overflow and undersupply. This point will be the optimum decision to achieve the project objectives.

3.2 Results

With the alternatives checked and the optimum decisions found, I would proceed to explain the process of how I found results for each point on the grid, and justify why this method is accurate to produce the optimum. I would also explain how I ran the model multiple times for each combination to ensure that there is enough data to verify the decision criteria produce consistent results. For example, if I had determined that the combination of 45 docks and 20 initial bikes at the station was the optimum, which produced a minimum overflow of zero and a minimum undersupply of one, I would explain that this combination yielded in the lowest total overflow and undersupply, making it the most optimum point on the decision variable grid. Interpreting these results is fairly straightforward; by having 45 docks at the station and an initial bike amount of 20, it will produce the most optimum results to maximize customer satisfaction and minimize CitiBike costs. I would mention that although this model is not the best interpretation of the real-life system (as determined by the validation phase), it can still provide useful insight for speculation, especially since the weekend hours between 12-4pm seem to be among the busiest, meaning that with these values it is likely that the station will never overflow even during other days and times. Of course, a disclaimer that a more robust model must be created to truly verify these speculations. All of this would be written as if I were directly talking to CitiBike representatives to offer them advice on how to improve their system - in other words as if I were making personal recommendations to someone.

4. Discussion & Conclusion

As a result of the lack of a functional model, I am unable to give final remarks on the results. However, if I did have results, I would reiterate that they are likely not very useful because the model does not take into account many important factors. The results may offer speculative insight in which actions to pursue to optimize the system, but it still remains to be speculative and I personally would not recommend taking action on them. In order to create a model that will produce useful and accurate results, the model must be built more robust by considering the many other important factors and decision variables to minimize overflow and undersupply, among other important criteria such as oversupply and transportation costs. As the model currently stands, it does not produce significant and actionable results.

For the future, I would consider including additional data that CitiBike provides, such as bike type (electric or normal), member type (casual or member), and the many other stations. I would also deploy field studies to determine how customers react when they are faced with overflow or undersupply, and then apply that data into a reneging function. I would also include data across all hours and days of the week, since the CitiBike stations are 24/7 operational. There are also cases when bikes are broken and require maintenance, which can be included in the model to determine the costs.

4.1 Personal Remarks

Overall, I really enjoyed trying to complete this assignment. Even though I was unable to produce the results I yearned for, I still learned a lot, particularly about processing and preparing data in a manner that will be easy and useful for analysis later down the road. I have used PANDAS and other data repositories to create an analysis report in the past, but never with such extensive and detailed data as the CitiBike repository (my laptop severely struggled to visualize 2.3+ million data points). Being able to recognize which data to cut out took a surprising amount of time because it was a very iterative process. I would start by filtering some data out, continue on with the project, then realize that I needed to make changes by filtering more or adding back certain data. After many hours, though, I had become very familiar with the data I was working with and had a clear vision of how to work with the data to produce useful results. Even with such a clear understanding of my data, I still found myself hitting crossroads and spending many more hours trying to pass them. For example, I had spent many hours testing to find the best way to filter the data I needed by days and by hours. I ended up having to create an incredibly ugly block of code that broke down the dataframes into individual days, which then broke down into individual hours (this took MANY lines of code). Then I selected which hours I specifically wanted, and concatenated back all the data into a single dataframe per day. I'm certain it wasn't the most optimal way of doing this, but I felt an indescribable amount of joy when it worked (such is the life of a researcher I'm sure). Another example was trying to splice the date/time data to find interarrival times. I found some online references on the DateTime library and spent a few hours trying to understand the documentation and how to properly implement it in my model. As far as creating the discrete event model, I was only able to successfully create the arrival and return generators, and display those time sequences in the environment with the four hour timespan. However, as soon as I tried to incorporate the other functions for the bikes, docks, transition, and negging, I could not seem to get it to work despite multiple hours of trying. Trust me, I am far more disappointed with this outcome than you are. Up to this point, I had spent roughly 25 hours creating the work, which to be honest sounds a little embarrassing considering how little I seem to have accomplished. As I said in the report, it was an incredibly busy few weeks for me with finals, reports, and presentations, so unfortunately I was unable to spend more time early on.

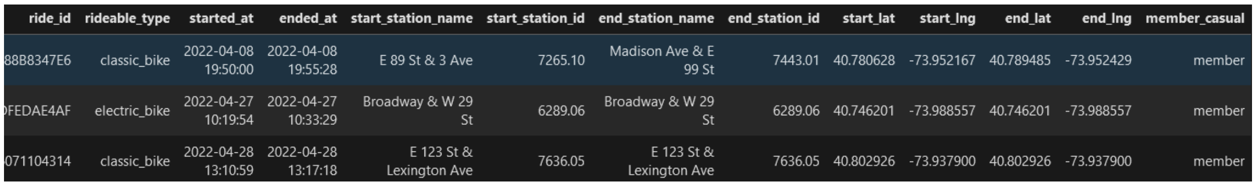
5. Sources

[1] - Full Repository || Zip download

[2] - Stack Overflow DateTime library reference

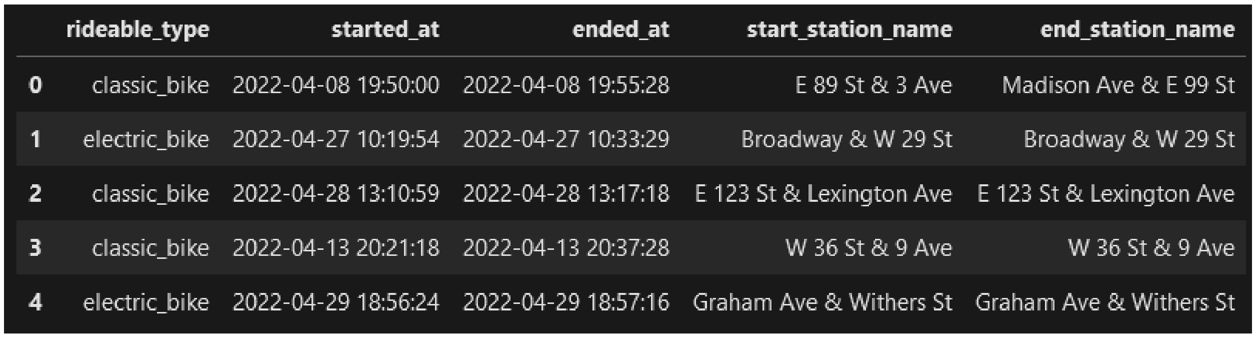
6. Appendix

1 - First three data entries of unaltered dataframe



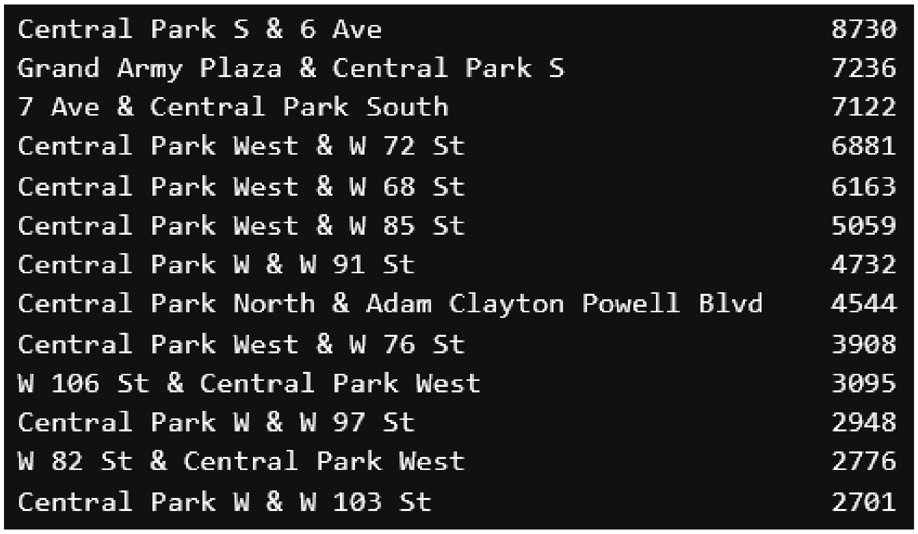
ride_id	rideable_type	started_at	ended_at	start_station_name	start_station_id	end_station_name	end_station_id	start_lat	start_lng	end_lat	end_lng	member_casual
88B8347E6	classic_bike	2022-04-08 19:50:00	2022-04-08 19:55:28	E 89 St & 3 Ave	7265.10	Madison Ave & E 99 St	7443.01	40.780628	-73.952167	40.789485	-73.952429	member
FEDAE4AF	electric_bike	2022-04-27 10:19:54	2022-04-27 10:33:29	Broadway & W 29 St	6289.06	Broadway & W 29 St	6289.06	40.746201	-73.988557	40.746201	-73.988557	member
071104314	classic_bike	2022-04-28 13:10:59	2022-04-28 13:17:18	E 123 St & Lexington Ave	7636.05	E 123 St & Lexington Ave	7636.05	40.802926	-73.937900	40.802926	-73.937900	member

2 - First five data entries with filtered columns



	rideable_type	started_at	ended_at	start_station_name	end_station_name
0	classic_bike	2022-04-08 19:50:00	2022-04-08 19:55:28	E 89 St & 3 Ave	Madison Ave & E 99 St
1	electric_bike	2022-04-27 10:19:54	2022-04-27 10:33:29	Broadway & W 29 St	Broadway & W 29 St
2	classic_bike	2022-04-28 13:10:59	2022-04-28 13:17:18	E 123 St & Lexington Ave	E 123 St & Lexington Ave
3	classic_bike	2022-04-13 20:21:18	2022-04-13 20:37:28	W 36 St & 9 Ave	W 36 St & 9 Ave
4	electric_bike	2022-04-29 18:56:24	2022-04-29 18:57:16	Graham Ave & Withers St	Graham Ave & Withers St

3 - Value Counts of all stations with “Central Park”



```
Central Park S & 6 Ave                         8730
Grand Army Plaza & Central Park S              7236
7 Ave & Central Park South                     7122
Central Park West & W 72 St                    6881
Central Park West & W 68 St                    6163
Central Park West & W 85 St                    5059
Central Park W & W 91 St                       4732
Central Park North & Adam Clayton Powell Blvd  4544
Central Park West & W 76 St                    3908
W 106 St & Central Park West                   3095
Central Park W & W 97 St                       2948
W 82 St & Central Park West                    2776
Central Park W & W 103 St                      2701
```