Train Delay Analysis in New York City

Victor Kuang

December 2022

1 Abstract

The New York City Subway system provides service to millions a day but is prone to extremely costly delays that can greatly impact the city. In hopes of making delays easier to visualize, this paper wishes to analyze delays over the most recent one year period the MTA Dashboard has available using a random delay model and a probability distribution model.

2 Introduction

The New York City Subway line is one of the busiest subway lines in the world. With 1.6-3 million rides a day, it is an important facet in the lives of the city's 8 million residents. Despite having such a crucial role in the life of the city, it does not always run smoothly and often times runs into delays. For instance, out of the 6891 Q trains that were scheduled in July 2021, 1723 of those trains were delayed. This amount of delay is simply a fraction of the 36 lines that run in the city, which can cost up to $389 million dollars a year.

To help understand these delays, this paper will try to analyze delays from the July 2021 to July 2022 cycle, the most recent full year's worth of data available according to the MTA Subway Dashboards. In understanding a year's worth of delay, this paper hopes to give a hypothetical model for delay expectation. For convenience, only results on the Q train will be discussed.

3 Methods

As this was to be a delay analysis from July 2021 to July 2022, typical train delay analysis methods using an API do not work well in this project because the wanted data is not real time. Unfortunately, the MTA does not have historical data available to this point, which does not allow for more 'realistic' testing due to external factors such as weather. To compensate for this, we will be analyzing two methods of train delay prediction: randomly generated delays, and probability distributions.

3.1 Calculation Explanation

Using the MTA Subway Performance Dashboard, data can be found regarding the amount of trains scheduled in a month can be found. Using the terminal on time performance, which is the percentage of trains that arrive to their destination on time, the formula to find the amount of delays in a month is the following:

$$D = S \times (1 - T)$$

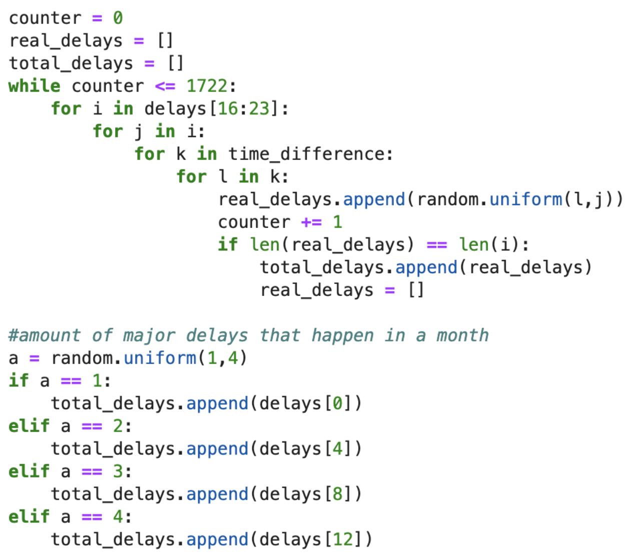
Where D is the amount of delayed trains, S is the amount of scheduled trains, and T is the terminal on time performance. For the Q train, the values are as follows:

Month	S	T	D
July 2021	6891	0.75	1723
August 2021	6923	0.708	2022
September 2021	6357	0.708	1856
October 2021	6615	0.76	1589
November 2021	6615	0.759	1594
December 2021	6615	0.726	1813
January 2022	6630	0.765	1558
February 2022	5977	0.761	1429
March 2022	7241	0.757	1760
April 2022	6615	0.79	1390
May 2022	6891	0.772	1571
June 2022	7496	0.773	1701
July 2022	6853	0.728	1864

In addition to this, the delays also has references to the NYC Comptroller's data providing major delays to be anything that makes wait time longer than 2, medium delays to be between 1.5-2 times longer wait times, and minor delays to be between 1.25-1.5 times longer wait times.

3.2 Random Delay Model

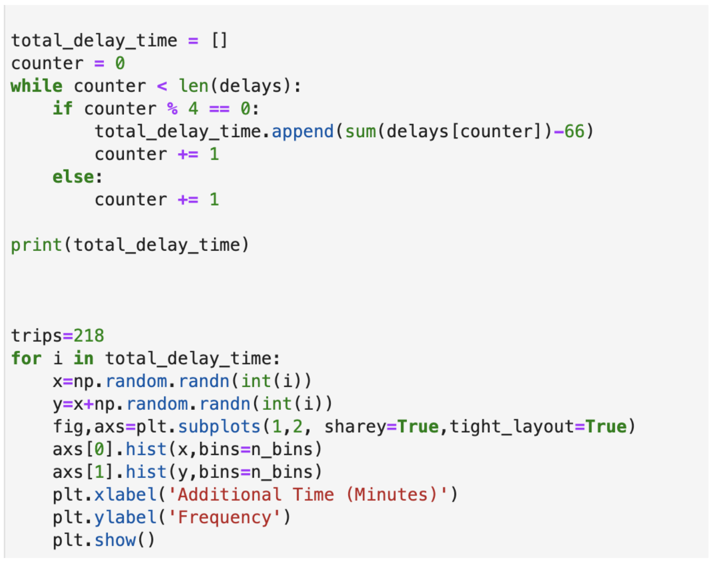
Using the data above, randomly generated delays were created based on the amount of delays that happened. The code is shown on figure 1. The idea was the plot the wait times by a monthly basis to analyze the delays.



```python
counter = 0
real_delays = []
total_delays = []
while counter <= 1722:
    for i in delays[16:23]:
        for j in i:
            for k in time_difference:
                for l in k:
                    real_delays.append(random.uniform(l,j))
                    counter += 1
                    if len(real_delays) == len(i):
                        total_delays.append(real_delays)
                        real_delays = []

#amount of major delays that happen in a month
a = random.uniform(1,4)
if a == 1:
    total_delays.append(delays[0])
elif a == 2:
    total_delays.append(delays[4])
elif a == 3:
    total_delays.append(delays[8])
elif a == 4:
    total_delays.append(delays[12])
```

Figure 1: Code of the random delay model



```python
total_delay_time = []
counter = 0
while counter < len(delays):
    if counter % 4 == 0:
        total_delay_time.append(sum(delays[counter])-66)
        counter += 1
    else:
        counter += 1

print(total_delay_time)



trips=218
for i in total_delay_time:
    x=np.random.randn(int(i))
    y=x+np.random.randn(int(i))
    fig,axs=plt.subplots(1,2, sharey=True,tight_layout=True)
    axs[0].hist(x,bins=n_bins)
    axs[1].hist(y,bins=n_bins)
    plt.xlabel('Additional Time (Minutes)')
    plt.ylabel('Frequency')
    plt.show()
```

Figure 2: Code of the distribution

3.3 Probability Distribution Model

As an alternative to the random delay model, the probability distribution model instead analyzes the probability that delays happen. In this method, rather than comparing hypothetical delay times with expected waiting times, this methods aims to analyze the spread of train delays to produce a more 'cumulative' model. The code is on figure 2. The idea was supposed to add individual times along a path up to get a delay time.

4 Results

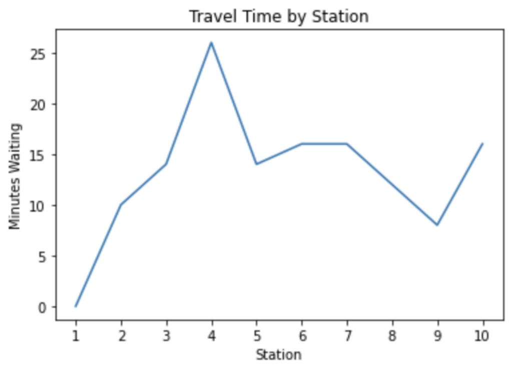


Figure 3: Plot of normal trip. X axis is the station name and Y axis is the minutes spent waiting

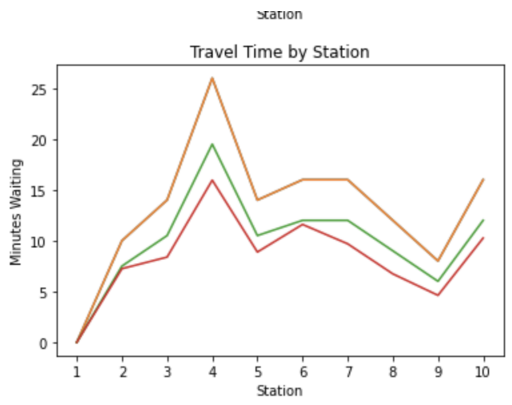


Figure 4: Code of the random delay model. Same axes as before with orange as a 4x delay, green being 2x, and red being an artificial minor delay.

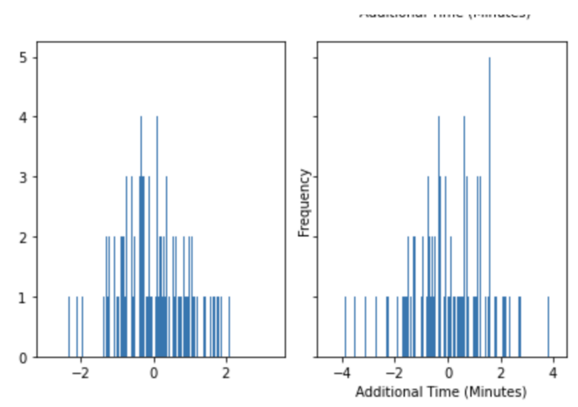


Figure 5: Distribution of major delays

4.1 Random Delay Model

The random delay model turns out to not be as dynamic as wanted, as shown on figures 3 and 4. Although it does show case different delay times, the result appears to be graphs that look as if the train delayed just as it reached its destination despite having an overall smooth train ride.

4.2 Probability Distribution Model

In an attempt to fix the random delay model, this tried to distribute the given train data to make a more realistic model, where train delays are usually a result of accumulation rather than sudden events. These distributions, shown in figure 5, show the frequency of events based on a given delay benchmark. The X axis allows for a small range where 0 is the delay being analyzed and the Y axis is how frequently it would be. Unfortunately, this does not seem to have the end goal of adding times to get a final delay time, as this distribution is very centered around the benchmarks from the Comptroller.

4.3 Areas of Improvement

To best analyze data over a year, it would be best to record historical data over a year to better analyze delays and make analysis off of that given data. The probability distribution model, while tried to fix the random delay model, does not seem to allow for a more dynamic analysis with more factors.

5 Conclusion

This paper analyzed delay results on the Q train using two methods. The randomization tried to generate delays according to how many delays there were in a month, whereas the distribution method found how frequent a delay would be around a given benchmark. The randomization methods was successful in creating hypothetical delayed times but fell short of appearing less linear. The distribution method was a step forward in introducing a more dynamic environment. Further improvements could be used with real data taken over a year to allow for more factors and more accurate analysis.

6 Appendix

1. (Comptroller.nyc.gov)

2. "Subway Dashboard." Subway Dashboard, http://dashboard.mta.info/.

3. Wang, Pu, and Qing-peng Zhang. "Train Delay Analysis and Prediction Based on Big Data Fusion." Transportation Safety and Environment, vol. 1, no. 1, 2019, pp. 79–88., https://doi.org/10.1093/tse/tdy001.

4. Yang, Yuxiang, et al. "Statistical Delay Distribution Analysis on High-Speed Railway Trains - Railway Engineering Science." SpringerLink, Springer Singapore, 13 June 2019, https://link.springer.com/article/10.1007/s40534-019-0188-z.