Visualizing NJ Transit Data

Tayo Ilunga-Reed

The Dataset

- 27 months of NJT and Amtrak arrival/departure data 3/2019-5/2020
- Amtrak data limited
- 1000 trains/day, 167 stations

- 11 columns: date, route number, stop sequence, origin/destination name and ID, scheduled and actual time, delays

date	train_id	stop_sequen	from	from_id	to	to_id	scheduled_ti	actual_time	delay_minut	status	line	type
3/1/18	3805	1	New York Pe	105	New York Pe	105	3/2/18 1:22	3/2/18 1:21	0	departed	Northeast Cc	NJ Transit
3/1/18	3805	2	New York Pe	105	Secaucus Up	38187	3/2/18 1:31	3/2/18 1:31	0.13333333	departed	Northeast Cc	NJ Transit
3/1/18	3805	3	Secaucus Up	38187	Newark Penr	107	3/2/18 1:40	3/2/18 1:40	0.11666667	departed	Northeast Cc	NJ Transit
3/1/18	3805	4	Newark Penr	107	Newark Airp	37953	3/2/18 1:45	3/2/18 1:45	0.16666667	departed	Northeast Cc	NJ Transit
3/1/18	3805	5	Newark Airp	37953	North Elizabe	109	3/2/18 1:49	3/2/18 1:49	0.16666667	departed	Northeast Cc	NJ Transit
3/1/18	3805	6	North Elizabe	109	Elizabeth	41	3/2/18 1:52	3/2/18 1:52	0.01666667	departed	Northeast Cc	NJ Transit
3/1/18	3805	7	Elizabeth	41	Linden	70	3/2/18 1:58	3/2/18 1:58	0.08333333	departed	Northeast Cc	NJ Transit
3/1/18	3805	8	Linden	70	Rahway	127	3/2/18 2:02	3/2/18 2:01	0	departed	Northeast Cc	NJ Transit
3/1/18	3805	9	Rahway	127	Metropark	83	3/2/18 2:08	3/2/18 2:08	0	departed	Northeast Cc	NJ Transit
3/1/18	3805	10	Metropark	83	Metuchen	84	3/2/18 2:13	3/2/18 2:13	0.16666667	departed	Northeast Cc	NJ Transit
3/1/18	3805	11	Metuchen	84	Edison	38	3/2/18 2:18	3/2/18 2:18	0.15	departed	Northeast Cc	NJ Transit
3/1/18	3805	12	Edison	38	New Brunsw	103	3/2/18 2:23	3/2/18 2:23	0.15	departed	Northeast Cc	NJ Transit
3/1/18	3805	13	New Brunsw	103	Jersey Avenu	32906	3/2/18 2:28	3/2/18 2:28	0.13333333	departed	Northeast Cc	NJ Transit
3/1/18	3805	14	Jersey Avenu	32906	Princeton Jur	125	3/2/18 2:42	3/2/18 2:33	0	departed	Northeast Cc	NJ Transit
3/1/18	3805	15	Princeton Jur	125	Hamilton	32905	3/2/18 2:49	3/2/18 2:39	0	departed	Northeast Cc	NJ Transit
3/1/18	3805	16	Hamilton	32905	Trenton	148	3/2/18 3:06	3/2/18 2:42	0	estimated	Northeast Cc	NJ Transit
3/1/18	2312	1	Bay Head	13	Bay Head	13	##########	##########	1.05	departed	No Jersey Co	NJ Transit
3/1/18	2312	2	Bay Head	13	Point Pleasa	122	##########	##########	3.05	departed	No Jersey Co	NJ Transit
3/1/18	2312	3	Point Pleasa	122	Manasquan	79	##########	##########	3.16666667	departed	No Jersey Co	NJ Transit
3/1/18	2312	4	Manasquan	79	Spring Lake	141	##					Transit
3/1/18	2312	5	Spring Lake	141	Belmar	15	##					Transit
3/1/18	2312	6	Belmar	15	Bradley Beac	22	##					Transit
3/1/18	2312	7	Bradley Beac	22	Asbury Park	8	##					Transit
3/1/18	2312	8	Asbury Park	8	Allenhurst	4	##					Transit
3/1/18	2312	9	Allenhurst	4	Elberon	40	##					Transit

Research Question

What are the best and worst performing stations, how late can I be for *x* train at *y* station?

Organizing the Data

- Use a massive list to quickly scrape all csv files
- With months specified, for loops can append the data
- Create dataframe from the list
- Make a smaller frame with one day's data for easier sampling/runtime

```
#packages
import pandas as pd
import csv
#setting up the data
years = ["2019","2020"]
months_2018 = ["03","04","05","06","07","08","09","10","11","12"]
months_2019 = ["01","02","03","04","05","06","07","08","09","10","11","1
months_2020 = ["01","02","03","04","05"]


df = []
#read through files of all months of each year and append them to df
for j in months_2018:
    df1 = pd.read_csv("2018_"+j+".csv", index_col=None)
    df.append(df1)
for j in months_2019:
    df1 = pd.read_csv("2019_"+j+".csv", index_col=None)
    df.append(df1)
for j in months_2020:
    df1 = pd.read_csv("2020_"+j+".csv", index_col=None)
    df.append(df1)
```

```
#turn df1 into a dataframe (df2)
df2 = pd.DataFrame()
for i in range(len(df)):
    df2 = pd.concat([df2, df[i]])

#make a smaller dataframe for one day to speed things up/find examples
df3 = df2[(df2['date'] == "2018-03-01")]
```

Generating a Timetable

- Make an empty list for all train_ids
- Use the unique() function to isolate all the train numbers
- Run a for loop through the smaller dataframe to populate origins and destinations list, then print the all the values together to see all routes

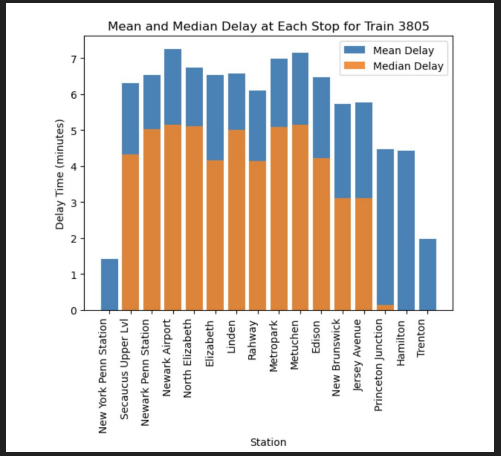
```
#creating a list of all the routes
#make three separate lists based on dataframe columns: train_id number, first and last stop
train_ids = []
train_ids = df3['train_id'].unique()
origins = []
destinations = []


#use for loop to sort trains (first cell in 3rd column = origin, last cell = destination)
for i in train_ids:
    temp = df3[(df3['train_id'] == i)]
    origins.append(df3[(df3['train_id'] == i)].iloc[0,3])
    destinations.append(df3[(df3['train_id'] == i)].iloc[len(temp)-1,3])

#print all the trains as they appear with first and last stops (842 trains on 3/1/18 alone)
for i in range(len(train_ids)):
    print(f"Train No.: {train_ids[i]}. From: {origins[i]} To: {destinations[i]}")
```

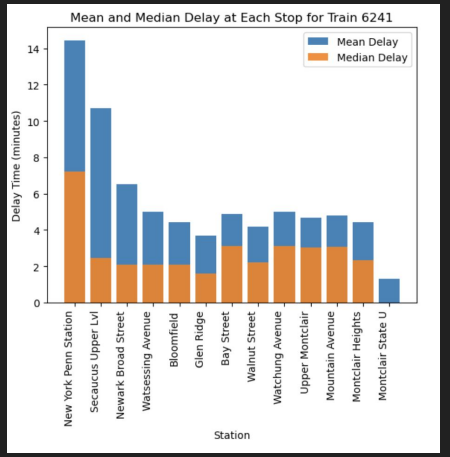
```
Train No.: 3809. From: New York Penn Station To: Hamilton
Train No.: 1134. From: Suffern To: Secaucus Lower Lvl
Train No.: 3831. From: New York Penn Station To: Hamilton
Train No.: 3200. From: Long Branch To: Secaucus Upper Lvl
Train No.: 0643. From: Hoboken To: Denville
Train No.: 3209. From: New York Penn Station To: Little Silver
Train No.: 3236. From: Long Branch To: Secaucus Upper Lvl
Train No.: 3838. From: Trenton To: Secaucus Upper Lvl
Train No.: A170. From: Philadelphia To: Newark Penn Station
Train No.: 1105. From: Hoboken To: Mahwah
Train No.: 6320. From: Summit To: Newark Broad Street
Train No.: 0211. From: Hoboken To: Montclair Heights
Train No.: 0447. From: Hoboken To: Peapack
Train No.: A184. From: Philadelphia To: Newark Penn Station
Train No.: A148. From: Philadelphia To: Newark Penn Station
Train No.: 3800. From: Trenton To: Secaucus Upper Lvl
```

Mean and Median Delays Across a Route's Stations

- Use input to identify a route, then isolate its data in a dataframe
- Filter the dataframe by stop sequence to get all stations in order
- Use mean() and median() functions to calculate values
- Their distribution will show the skew caused by extreme delays
- Plot the values in matplotlib
- Tremendous variation between routes

```
#make a new dataframe that isolates one route
train_id = input("What's your train's route number?")
filtered_df = df2[df2['train_id'] == train_id]
#sort it by station
grouped_df = filtered_df.groupby('stop_sequence')

#take the average of each station's total delays
average_delays = grouped_df['delay_minutes'].mean()

#set up bar graph showing mean
plt.figure(figsize=(10,2))
plt.bar(average_delays.index, average_delays)
plt.xlabel('Station')
plt.xticks(average_delays.index, filtered_df.groupby('stop_sequence')['to'].first(), rotation=90, ha='right')
plt.ylabel('Average Delay (minutes)')
plt.title(f'Average Delay at Each Stop for Train {train_id}')
plt.show()

#makea median with the same grouped dataframe
median_delays = grouped_df['delay_minutes'].median()
```

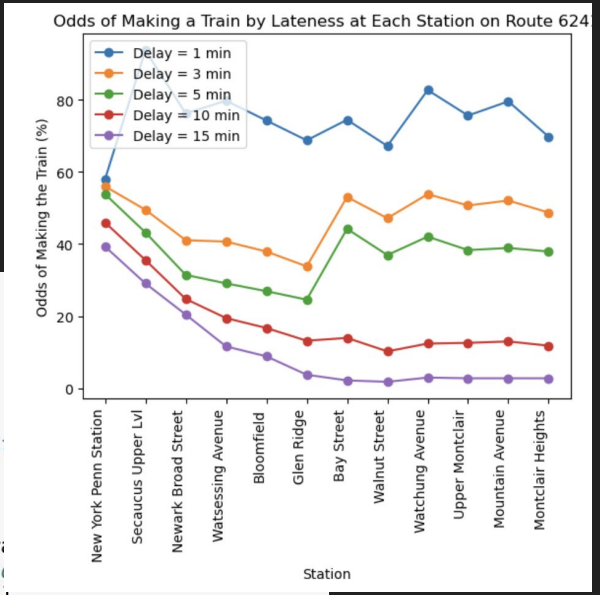




Straggler's Guide: Probabilities at Lateness Intervals

- Set delay intervals, then iterate through a trains arrival at all stops, dividing delays that exceed interval value by total arrivals

```
#find the stations and set delay intervals
stations = train_frame1['from'].unique()
delay_times = [1, 3, 5, 10, 15]
#make another dataframe with the stations and intervals
probabilities_df = pd.DataFrame(index=stations, columns=delay_times)

#use a for loop to create a dataframe for each time a certain train leaves a certain s
for delay_time in delay_times:
    probabilities = []
    for station in stations:
        train_frame2 = train_frame1[(train_frame1['train_id'] == train_id) & (train_fra
        #sum the times the train's delay exceeds the interval and divide them by the t
        prob = sum(train_frame2['delay_minutes'] >= delay_time) / train_frame2.shape[0]
        probabilities.append(prob)
    #add the probabilities list to the corresponding column within probabilities dataframe
    probabilities_df[delay_time] = probabilities

#make it a percent
percentages_df = probabilities_df * 100
```

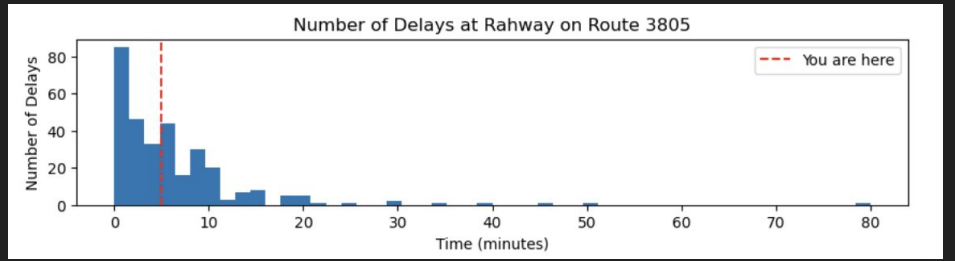


Straggler's Tool: What's My Chance of Making the Train?

- Takes inputs: train, station, lateness
- Returns probability and a histogram for visual aide
- Same probability technique matched to a specific station and interval

```
#get inputs for route, station, and lateness
train_id = input("What's your train route number?")
origin = input("What's your departure station?")
delay_time = int(input("Be honest, how late are you?"))

#filter a frame of that train approaching that station
train_frame = (df2[(df2['train_id'] == train_id)*(df2['to'] == origin)])

#find the same probability but just for this instance
prob = sum(train_frame['delay_minutes'] >= delay_time)/train_frame.shape[0]

#print it
print('You have a ' f"{prob:.2%}" ' chance of making the train.')


#make a histogram showing the delays with a semispecific bin
plt.hist(train_frame['delay_minutes'], bins = 50)
plt.xlabel('Time (minutes)')
plt.ylabel('Number of Delays')
plt.title(f'Number of Delays at {origin} on Route {train_id}')
#mark the user's lateness within the histogram to give them a false sense of security
plt.axvline(x=delay_time, color='red', linestyle='--', label=f'You are here')
plt.legend()
```

```
What's your train route number?3805
What's your departure station?Rahway
Be honest, how late are you?5
You have a 47.27% chance of making the train.
```



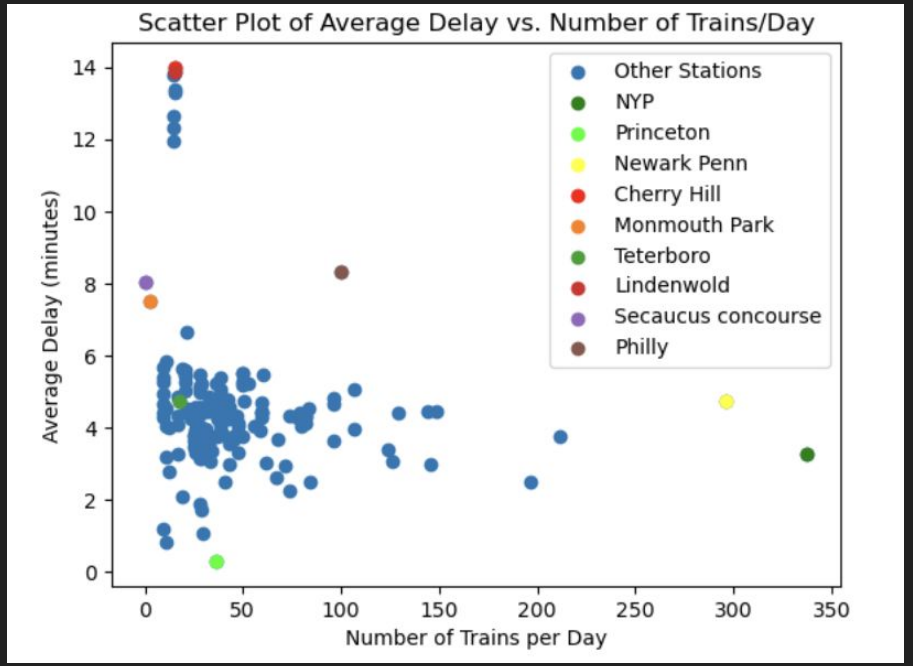
Analyzing Stations: Mean Delay vs. Trains per Day

- Estimating efficiency of stations
- Tracks could also be a good normalizing divisor
- Create dictionaries for with keys as stations, values as trains/day and mean delay time
- Plot them against each other in a scatterplot

```
#get a list of stations
stations = df2['from'].unique()
#make a dictionary
station_counts = {}
#get a day value to normalize the data
days = len(df2['date'].unique())


#make a for loop that rewrites the dictionary {station: #average number of trains/day}
for station in stations:
    count = df2['to'].eq(station).sum()/days
    station_counts[station] = count

#make another dictionary
station_delays = {}
#use another for loop to replicate the first dictionary but with average delay time
for station in stations:
    filtered_df = df2[df2['from'] == station]
    average_delay = filtered_df['delay_minutes'].mean()
    station_delays[station] = average_delay


#generate lists for #trains/day and average delay time to make graphing easier
num_trains = list(station_counts.values())
average_delays = list(station_delays.values())
```

Station Rankings

- Using trains per day as a normalizing factor
- Score: trains/day over delay time
  - Most Efficient: Princeton (125.3), NYP (103.6), Hoboken: (78.57)
  - Least Efficient: Secaucus Concourse (.003), Monmouth Park (.332), Egg Harbor City (1.078)




```
{'Princeton': 125.3277439639723,
'New York Penn Station': 103.59510716629495,
'Hoboken': 78.57349361177482,
'Newark Penn Station': 62.51654037380879,
'Secaucus Upper Lvl': 56.51273970966977,
'Trenton': 49.09908168155491,
'Princeton Junction': 41.23729308917686,
'Secaucus Lower Lvl': 36.632894312724225,
'Hamilton': 33.81385488427062,
```

```
{'Secaucus Concourse': 0.003373606773040421,
'Monmouth Park': 0.33252542887608966,
'Egg Harbor City': 1.0782390466001521,
'Cherry Hill': 1.090444262515596,
'Lindenwold': 1.1002345683025483,
'Atco': 1.1299311564172232,
'Pennsauken': 1.1409460870197219,
'Absecon': 1.173017792650664,
'Hammonton': 1.2050111239474026,
```

Tools Used

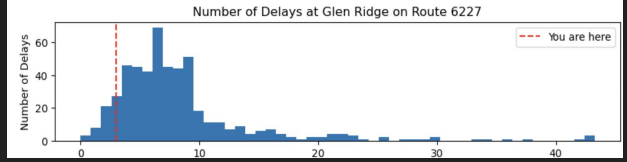
- Dataframes and their functions (unique(), mean(), median(), etc.)
- For loops to filter data, find matches between variables
- Lists to generate the original dataframe and present/compare data
- Dictionaries to create and compare key-value relationships
- Matplotlib for scatter plots, bar graphs, and histograms

Outputs

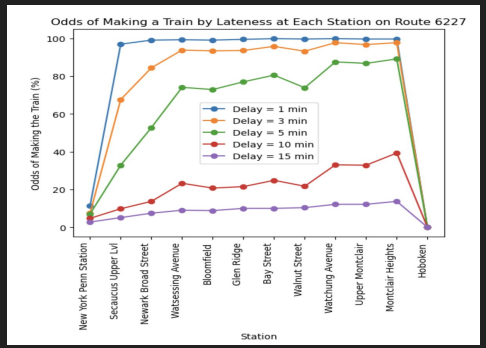
```
Train No.: 1207. From: Hoboken To: Ho-Ho-Kus
Train No.: 6210. From: Montclair State U To: Watsessing Avenue
Train No.: 4123. From: Princeton Junction To: Princeton Junction
Train No.: 6674. From: Dover To: Secaucus Upper Lvl
Train No.: 3937. From: New York Penn Station To: Hamilton
Train No.: 4124. From: Princeton To: Princeton
Train No.: 3930. From: Trenton To: Secaucus Upper Lvl
Train No.: 3595. From: New York Penn Station To: Perth Amboy
Train No.: 1652. From: New Bridge Landing To: Secaucus Lower Lvl
Train No.: 4112. From: Princeton To: Princeton
Train No.: A2175. From: New York Penn Station To: Trenton
Train No.: 4115. From: Princeton Junction To: Princeton Junction
Train No.: A2172. From: Philadelphia To: Newark Penn Station
```

Route List

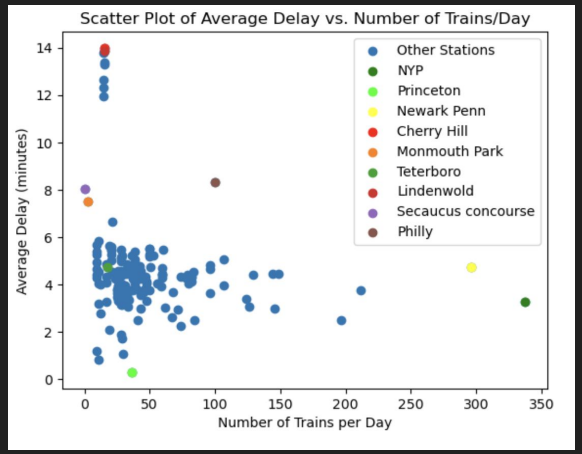
```
What's your train route number?6227
What's your departure station?Glen Ridge
Be honest, how late are you?3
You have a 93.55% chance of making the train.
```



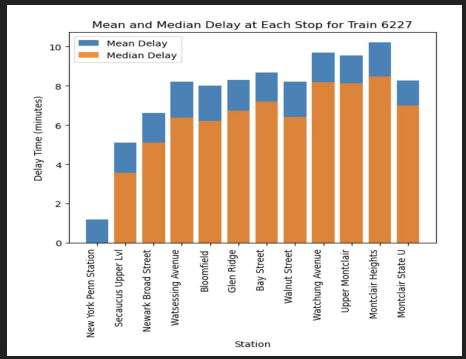
Probability Calculator



General Delay Estimator



Stations Scatterplot/Rankings



Mean/Median Bar Graphs

Conclusion

- Tremendous variability between routes and stations
- Biggest stations are optimized, account for minimal delays
  - They’re also often origin/terminus, which tend to have lower delay values
- Smaller stations on delay-plagued lines have the lowest scores
- Data issues: misreported delays, changes in stations/routes over time

Further Potential Research

- Analysis between different times and days: are trains slower at night or on weekends?
- Further data normalization: accounting for station size (number of tracks) and extreme delays (storms, accidents, etc.)
- Identifying most troubled routes, causes for delays
- Expanding the data: incorporating ridership
    - Do delays discourage ridership? Are the most popular trains the slowest?

Thanks for Listening!