Part 5 - SI 206 Final Report

Brian Metz, Dylan Shefman, Bobby Housel

Github Repository Link: https://github.com/dylanshefman/SI206FinalProject

In addition to your API activity results, you will be creating a report for your overall project. The report must include:

1. **The goal for your project**

The goal for our project was to gain a better understanding for how public transit systems in major US cities work alongside APIs and beautiful soup systems in order to record and make sense of their massive amounts of data. We selected Boston and Chicago as the two cities we would focus on (after finding that the Atlanta Transit API we had also originally selected was not working as we had anticipated). Our goal was to not only learn about how these major cities track their public transportation systems, but also to try and make sense of how efficient or inefficient these systems were. In both the Boston and Chicago APIs for public transportation, data is given indicating the predicted time of arrival and the actual time of arrival. Our main goal was to use the APIs alongside beautiful soups in order to compare total ridership in conjunction with how accurately predicted arrival/departure times were. We then wanted to double down and understand how this correlates with how many people ride each respective public transit line. Stated another way, we wanted to find out if there was a correlation between the difference in predicted versus actual arrival times, and if that affected how many people ride each respective line. Do travelers in major US cities optimize for taking the most consistent public transit route? Do travelers change which route they use depending on how consistently on time it is? We were determined to find out.

2. **The goals that were achieved**

Overall the entire project was a huge success. Our biggest failure, if you can call it that, was having to change our initial thesis and idea because the Atlanta API was not acting as we had hoped. Instead of using the Atlanta API in conjunction with Chicago and Boston, we ultimately wound up using a beautiful soup instead. Besides this failure, we were nearly successful in all the goals we set out to achieve. We were successful in pulling data from the Boston and Chicago APIs and our two beautiful soups. Both APIs acted as we had anticipated and provided us with data regarding **train offset arrival times** (**offset time** is the absolute value of the difference between predicted arrival time and actual arrival time). This gave us a look into the efficiency of each train line. In addition, we were successful in comparing the average total ridership versus average offset time. We were also successful in comparing relative stations to each other between Chicago and Boston, and how they each performed. The last goal we came up a bit short on, was seeing a stronger trend between ridership and offset times. We thought that there would be a strong correlation between offset arrival times and

ridership, but in fact there was not as much as we had anticipated. We believe this is likely due to the fact that there are many conditions that affect ridership other than only on time arrivals. These factors/variables could include elements such as how many stops the train line has and where it services. Despite our original hypothesis being slightly incorrect, we were still successful in interpreting the data between these variables and making sense of it.

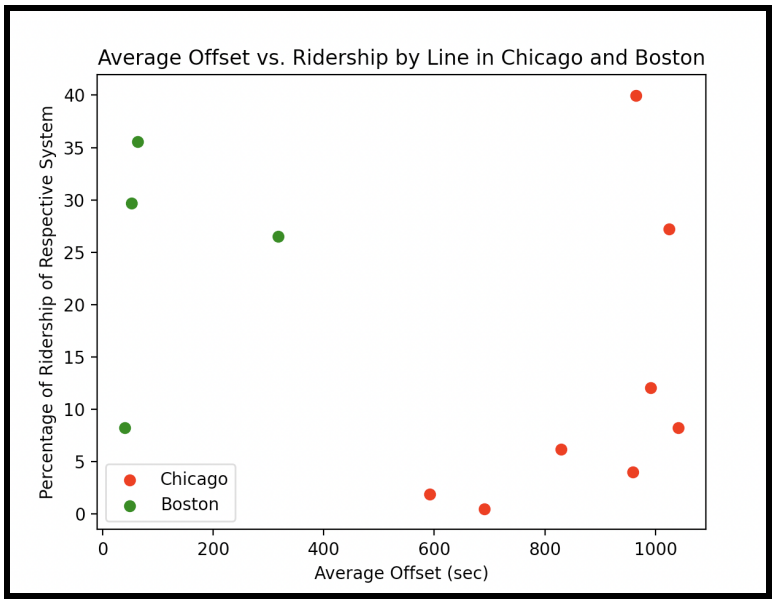
3. **The problems that you faced**

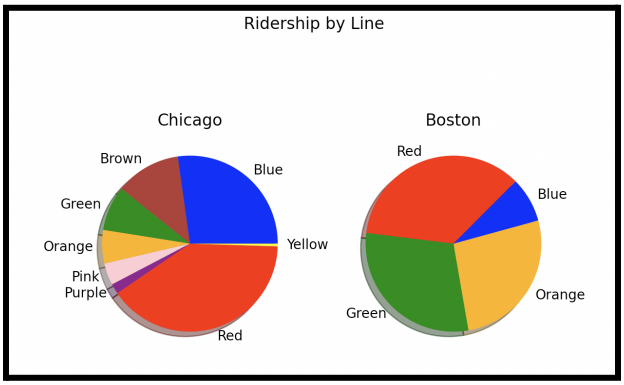
The first major problem that we faced was selecting the thesis of our project. When browsing Github and the internet looking for potential APIs to use, we were overwhelmed with the options. We found APIs ranging from describing the characters of our favorite TV shows all the way to tracking and recording religious affiliations in countries. Every time we thought we had an idea that might work, we would begin to play around with the APIs to soon discover they didn't behave as we thought. This led us right into our second problem that we faced. After finally thinking we had locked in our APIs and the thesis of our project, we began getting down to writing some code and messing around with the APIs. Just when we thought all the APIs we had selected were working appropriately, we realized that the API we had found for the Atlanta transit system did not function at all like we had thought and seemed to be very glitchy and unreliable. Because of that, we decided to rethink the thesis of our project and start using some beautiful soups instead of more APIs. The last problem we faced revolved around getting some of our tables to display correctly while using the beautiful soup we had selected. The beautiful soup website(s) was organized strangely and wasn't working as we had anticipated, so we wound up using a different site and ultimately resolved the problem. We ran into a handful of smaller problems and questions throughout the project, which we detail in our documentation of all resources we used (question #8 of our report). These challenges ranged from trying to figure out how to change colors of data in scatter plots, all the way to understanding how to interpret lingo and resources from the transit authority websites.

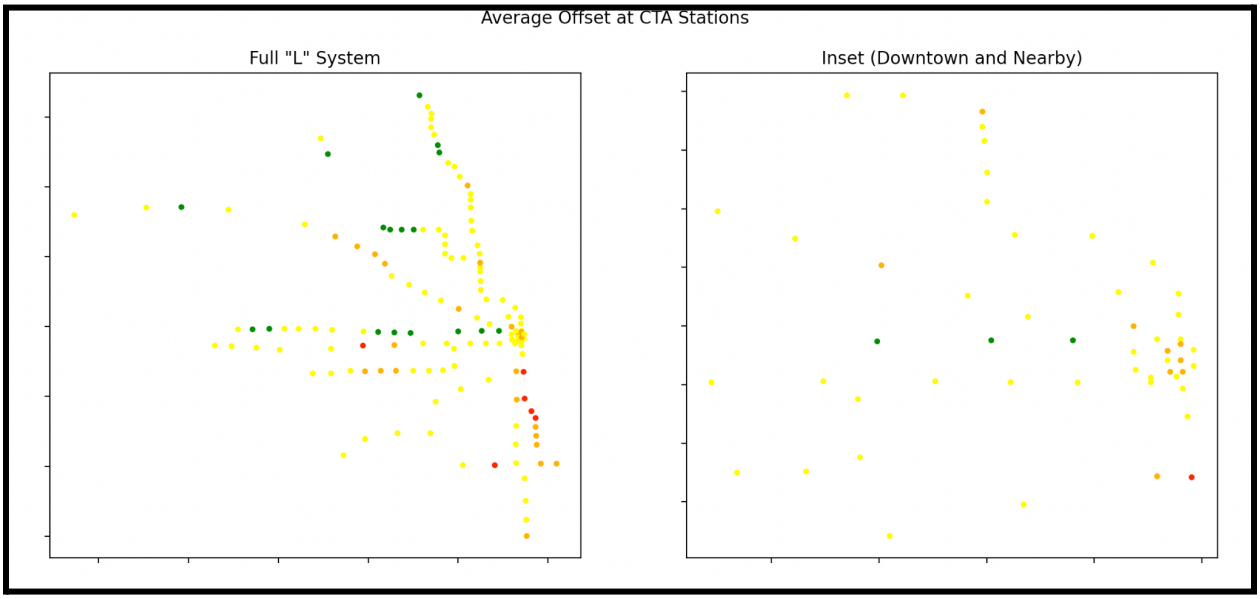
4. **Your file that contains the calculations from the data in the database**

https://drive.google.com/file/d/1cbELn-FSDUnMgoC-1MV0K6sPt2LnyuFx/view?usp=sharing

5. **The visualization that you created (i.e. screenshot or image file)**


Average Offset vs. Ridership by Line in Chicago and Boston
Percentage of Ridership of Respective System
Average Offset (sec)
Chicago
Boston
0
5
10
15
20
25
30
35
40
0
200
400
600
800
1000


Ridership by Line
Chicago
Brown
Blue
Green
Orange
Yellow
Pink
Purple
Red
Boston
Red
Blue
Green
Orange


Average Offset at CTA Stations
Full "L" System
Inset (Downtown and Nearby)

6. **Instructions for running your code**

In order to run this code, you first need to open all our files in VS Code (or a similar coding environment). From there you first need to run the functions regarding ridership (chicago_ridership.py, boston_ridership.py, chicago.py, boston.py, visualization.py). In order to run visualizations.py, once the first visualization pops up, you have to press “x” for the ensuing visualization to pop up, and so on until all visualizations have been viewed. After all visualizations have been viewed, you can close the files and VS Code (or other coding environment), and our entire project will have been successfully run and viewed.

7. **Documentation for each function that you wrote. This includes the input and output for each function**

File	Function	Input	Output
chicago_ridership.py	get_ridership()	none	List of line objects representing each cta line
chicago_ridership.py	write_line_id_table( )	Cur and conn	none
chicago_ridership.py	write_to_db()	Ridership list, cur and conn	none
boston_ridership.py	get_ridership()	none	List of line objects representing each cta line
boston_ridership.py	write_line_id_table( )	Cur and conn	none
boston_ridership.py	write_to_db()	Ridership list, cur and conn	none
chicago.py	condense()	day, hour, min, sec	Integer representing condensed time (secs since beginning of month)
chicago.py	deg_to_secs()	deg	seconds
chicago.py	get_stations()	None (reads from	List of tuples which

		a text file)	each represent one station
chicago.py	get_url()	station_id	url
chicago.py	get_timing_from_url()	Url, cur	List of tuples which each represent a single trains arrival at a single station
chicago.py	create_tables()	Cur and conn	none
chicago.py	write_timing_to_db()	Timing_tup, cur and conn	none
chicago.py	write_location_to_db()	Tup, cur and conn	none
boston.py	deg_to_secs()	deg	seconds
boston.py	get_stations_by_color()	none (reads from a text file)	A dictionary of dictionaries that each represent one destination of one line
boston.py	get_station_pairs()	Inner-dicts returned from previous function	List of tuples, each representing a pair of stations (from station and to station)
boston.py	get_url()	Station1, station2	url
boston.py	get_trips_from_url()	Url, cur	List of trip objects, each representing one trains journey between two stations
boston.py	create_table()	Cur and conn	none
boston.py	write_to_db()	Trip object, cur and conn	none
visualization.py	create_table()	Cur and conn	none
visualization.py	write_avgs()	Cur and conn	none

visualization.py	scatter()	cur	none
visualization.py	pie()	cur	none
visualization.py	map()	Cur and conn	none

8. **You must also clearly document all resource you used. The documentation should be of the following form (date, issue description, location of resource, result (did it solve the issue?))**

Date	**Issue Description**	**Location of Resources**	**Result (did it solve the issue?)**
11/30/22	We used this text file and the transit authority websites to get the station IDs for all the train stations in each city.	Stops.txt file https://www.transitchicago.com/ https://www.mbta.com/ -We got this from the CTA (Chicago Transit Authority) and the MBTA (Massachusetts Bay Transit Authority) websites respectively -Stops.txt is a file within the GTFS data specification. GTFS [General Transit Feed Specification] is the feed for the Chicago and Boston transits. GTFS is a standardized format	We successfully obtained station IDs and were able to compare them from relative cities and make sense of the respective data from each.

		that transit agencies use to publish scheduling and geographic information on a vehicle-specific basis.	
12/2	How to use Matplotlib?	https://matplotlib.org/stable/tutorials/introductory/pyplot.html	We used this site to learn how to better organize a figure, how to use subplots, how to add labels, how to rotate an axis of a pie chart.
12/3	What is GTFS and how did it work?	https://gtfs.org/	GTFS.org was incredibly helpful in better explaining how GTFS works and how to best work with it
12/6	Code taking too wrong to run, too many duplicate string data. What to do?	https://www.geeksforgeeks.org/how-to-find-duplicate-values-in-a-sql-table-using-python/	Geeks for Geeks website was incredibly helpful in aiding our understanding for how to deal with duplicate string data and eliminating it
11/28	Receive data relative to ridership usage on each line in each city	https://en.wikipedia.org/wiki/List_of_Chicago_%22L%22_stations https://en.wikipedia.org/wiki/Massachusetts_Bay_Transportation_Authority	We beautiful souped the necessary data in order to compare it to data collected from each of our APIs.
12/11	How to change colors of points on	https://stackoverflow.com/questions/41	We found which colors to use and

	scatter plot?	709257/how-to-change-the-plot-line-color-from-blue-to-black#:~:text=The%20usual%20way%20to%20set,explicitely%20stating%20the%20color%20argument	how to change them
12/11	How to get rid of axis labels?	https://www.tutorialspoint.com/hide-axis-values-but-keep-axis-tick-labels-in-matplotlib	Successfully hid excess numbers and labels