Pafnuty's Taxis: Modeling Taxi Cab Routes in NYC

Kaylee Burns, Gary Cheng, Mitas Ray

Introduction

We chose to model the routes of taxicab drivers in New York for a given time interval as random walks among geographic regions of the city. The probability of transitioning between states was calculated by counting the number of rides to other states and dividing by the total number of rides out of that state. In addition to transitional probabilities, we associate an average fare with each edge.

Markov chains are well suited to trends in taxi cab drivers' routes and can enlighten us about (1) the popularities of various pickup locations over time, (2) opportunities for taxicab drivers to increase their revenue, and (3) the trends in "migration" among customers. This investigation not only provides direct financial benefit to drivers but satisfies our curiosity about transportation trends.

Methods (theory and pseudocode)

All code for our project can be found in the zip file we turned in or in our github repo: https://github.com/garyxcheng/pafnutys-taxi; All actual code can primarily be found in the_business.ipynb, MarkovChain.py, and State.py. The data that we used can be found here: https://www.kaggle.com/nyctaxi/yellow-taxis

We started by creating a Markov Chain class that handles all of the high level calculations -- such as finding the invariant distribution and running random walk simulations -- and keeps track of various State objects. The State objects kept track of which data points were a part of the state and what the transition probability and expected fare is to another state. The pseudocode for these two class files can be found in MarkovChain_Pseudocode.py.

It is infeasible for each pickup/dropoff location to be it's own state, so we used k-means to generate states and group together these pickup/dropoff locations. Initially, we ran 1000 update cycles to generate states, but found that it saved far more time to update the k-mean centers until the L2 norm change in latitude and longitude of the k-mean centers between each iteration became less than some epsilon (in our calculations we set epsilon to be 1 x 10^-12).

However, even with k-means, a key question that needed to be answered was how many centers (k) to use. After doing online research, we settled on the Elbow method to find the optimal number k. The method plots sum of squared error (SSE), which is the sum of squared distances for each point to its closest center, with respect to the number of centers (k). By analyzing the graph, we pick the minimum k value where the SSE plateaus. The plateau is the balance point between too few centers that group large regions together and too many centers that cluster very small regions.

We began our implementation of the elbow method by writing a function that created markov chains generated with k-means with parameter ranging from k=5 to k=50. For each of these

markov chains, we found the SSE. We defined finding the SSE of markov chains clustered from k=5 to k=50 with 1000 random data points as one iteration. We did 20 iterations of this operation and averaged it to generate the Elbow plot.

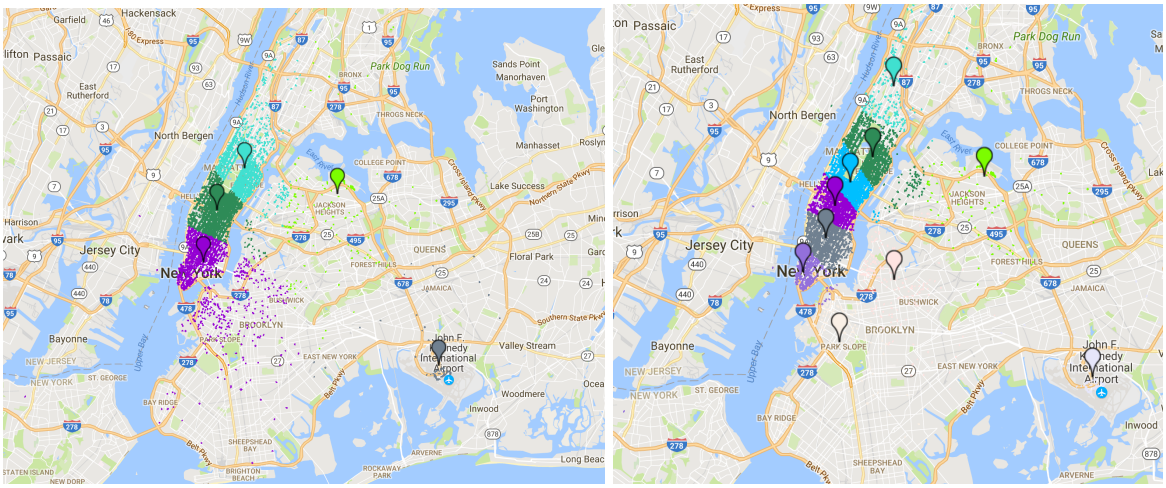
From the Markov Chains of variable state number, we generated a transition matrix whose invariant distribution is defined by the eigenvector with eigenvalue 1. We interpret the invariant distribution as our ranking of states in the PageRank algorithm. To visualize our invariant distributions, we created heatmaps with the gmplots library. The darkness of a spot on the map corresponds to the frequency of that coordinate in a given sequence of coordinates. Coordinate sequences were generated by duplicating cluster centers in proportion to their average number of visits in the steady state distribution.

Using our Markov Chains, we also sought to answer the question: “If I could only give n rides, which state should I start out in to generate the most money?” To realize this, we simulated a large number of random walks and averaged the profit collected along the walk. In addition to probabilities, edges of the Markov chain were assigned average fares, which were collected along the walk.

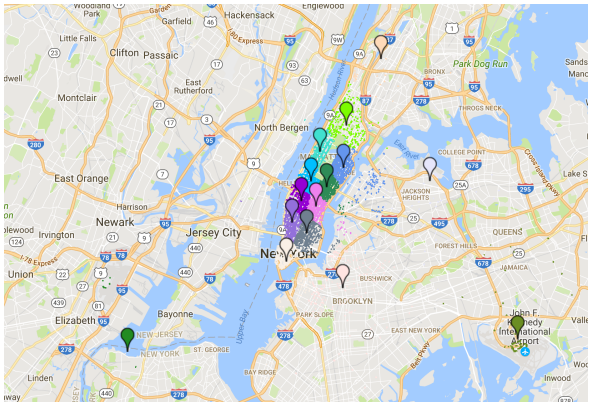
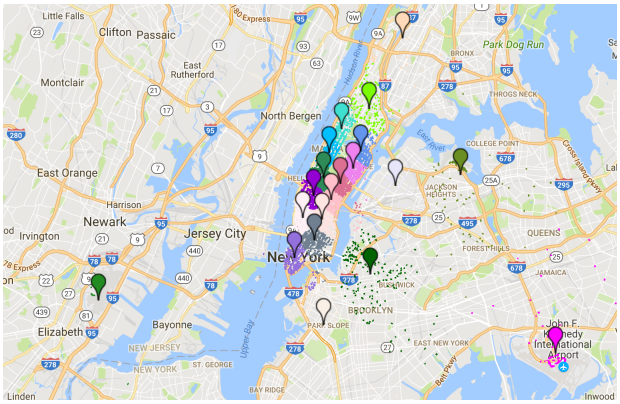
Experiments and Analysis

Modifying the value of K in K means.

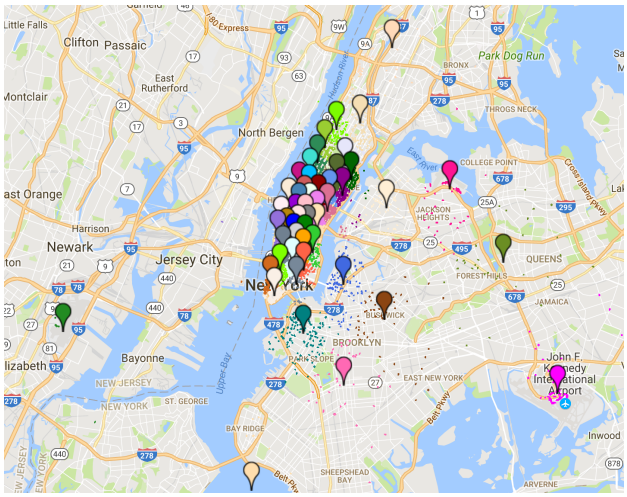
We wanted to explore how changing the number of states in our Markov Chain impacted clustering and the invariant distribution. We plotted diagrams of taxi cab pick up and drop off locations, colored by state, to visualize how our k-means algorithm broke the city down into k = 5, 10, 15, 20, and 50 geographic regions.



K = 5, K = 10

K = 15, K = 20

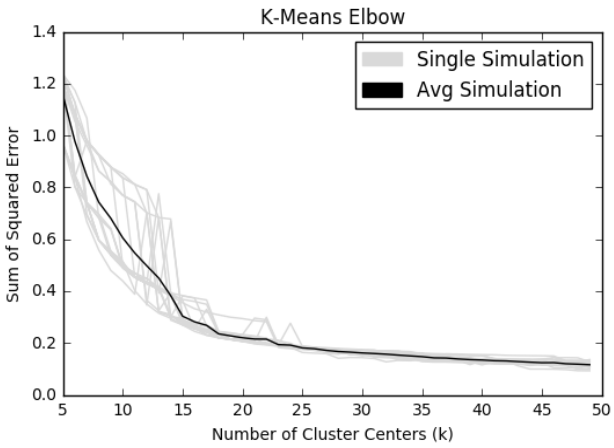


K = 50

Cluster centers arise in airports, like JFK and LaGuardia, as well as popular tourist locations, like Times Square or Rockefeller Center. Clusters on Manhattan are more densely distributed, while clusters in residential neighborhoods, like Brooklyn or the Bronx, are sparse and have less easily interpretable centers.

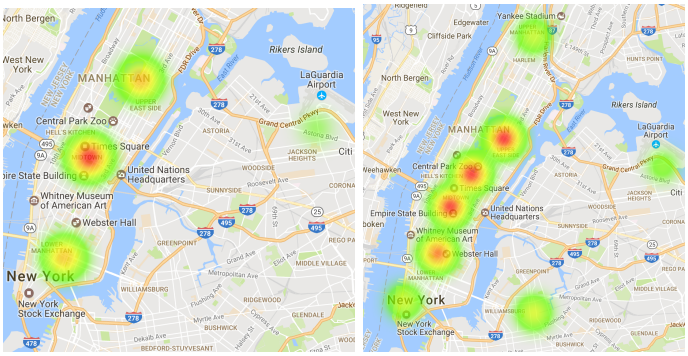
HTML files are included in the zip file. Readers wishing for a more interactive experience should refer to those for further exploration.

In addition to analysing the clusterings for a variable number of centers, we found the “optimal” value of k empirically by plotting the average sum of squared error. We selected a value at which the error began to plateau: k = 15.

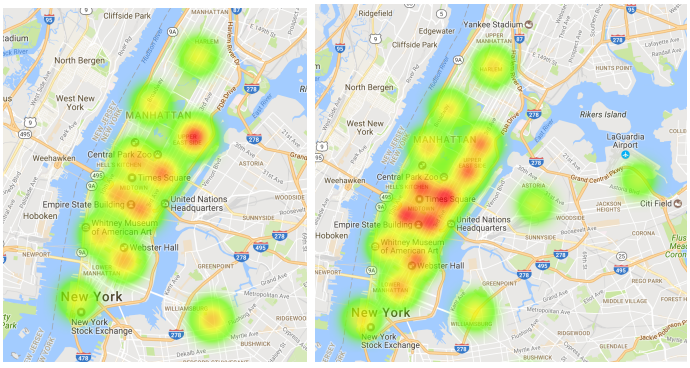


Invariant Distribution with Variable Number of States.

We used the invariant distribution of our markov chain as a proxy for the popularity of pickup and drop off locations, similar to the ranking of web pages in the the Pagerank algorithm. To visualize our ranking, we created heat maps of the cluster centers: clusters with a higher average number of visits at steady state are more red. Unsurprisingly, tourist attractions like Times Square are hot spots.



Closeup: heat maps of Manhattan for k = 5, k =10.



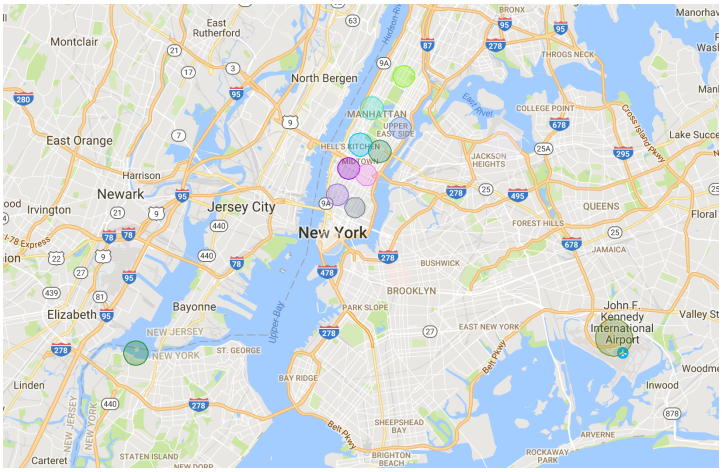
Closeup: heat maps of Manhattan for k = 15, k =20.

The specific values for the invariant distributions are presented in the attached jupyter notebook. We omitted them from our write up because the values of the invariant distribution are not ordered in an interpretable way and provide little information not captured in the above plots. Once again, the images' corresponding HTML files are available in the folder for a more interactive experience. Other map locations, notably airports like JFK, are not represented in the above images, but their steady state averages are also represented in the attached HTML file.

A final interesting result of our experiment was that we needed to double the amount of data we read in -- from 5,000 to 10,000 -- to get an irreducible Markov Chain for more than 10 cluster centers. The above heatmaps were generated on 10,000 sample points. The plot for 50 cluster centers was omitted because the chain was reducible for even 10,000 sample points.

Expected Revenue after 10 Rides.

We wanted to answer the question: If a taxi driver was to give 10 rides, in which state should they start to generate the most revenue? We calculated the expected amount of money that a taxi driver would make if they want from one state to another for each pair of states. We then made a random walk in the Markov chain for 100 simulations for each state.



The size of each dot in the above figure is proportional to the expected revenue after 10 rides starting from that state. Once again, specific values for each state are included in the jupyter notebook and an HTML file of the map corresponding to the image is in the folder.

Discussion/Limitations

Our main limitations were computation time and disk space. We wanted to work with the entire dataset, but restricted our analysis to the month of January instead. Incorporating a larger range of times would have allowed us to compare how Taxi cab travel changed throughout the year. We considered refining our model with a continuous time Markov Chain, but we encountered the concept after a bulk of the project was completed. We could also consider different times to compare taxi service at night life as compared to the daytime.