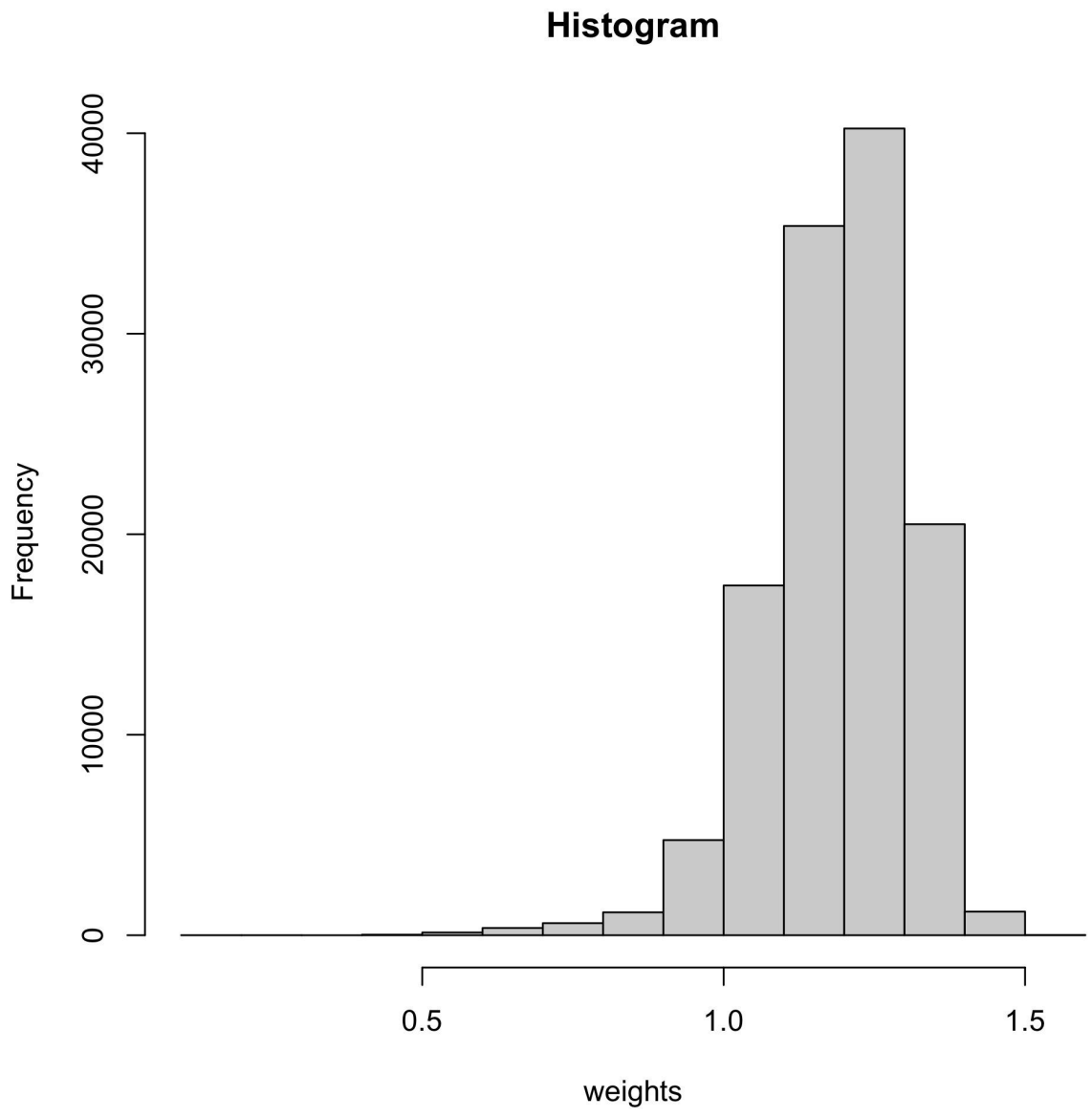
Problem 1

The bounds of correlation are [-1,1], where 0 denotes no relationship between i,j. -1 implies perfect anticorrelation and the opposite for 1. The log normalized return is advantageous as it imposes the normal distribution on the dataset while also forcing data to be positive. For stock prediction tasks, this is unavoidable as the value of a stock is bounded by 0.

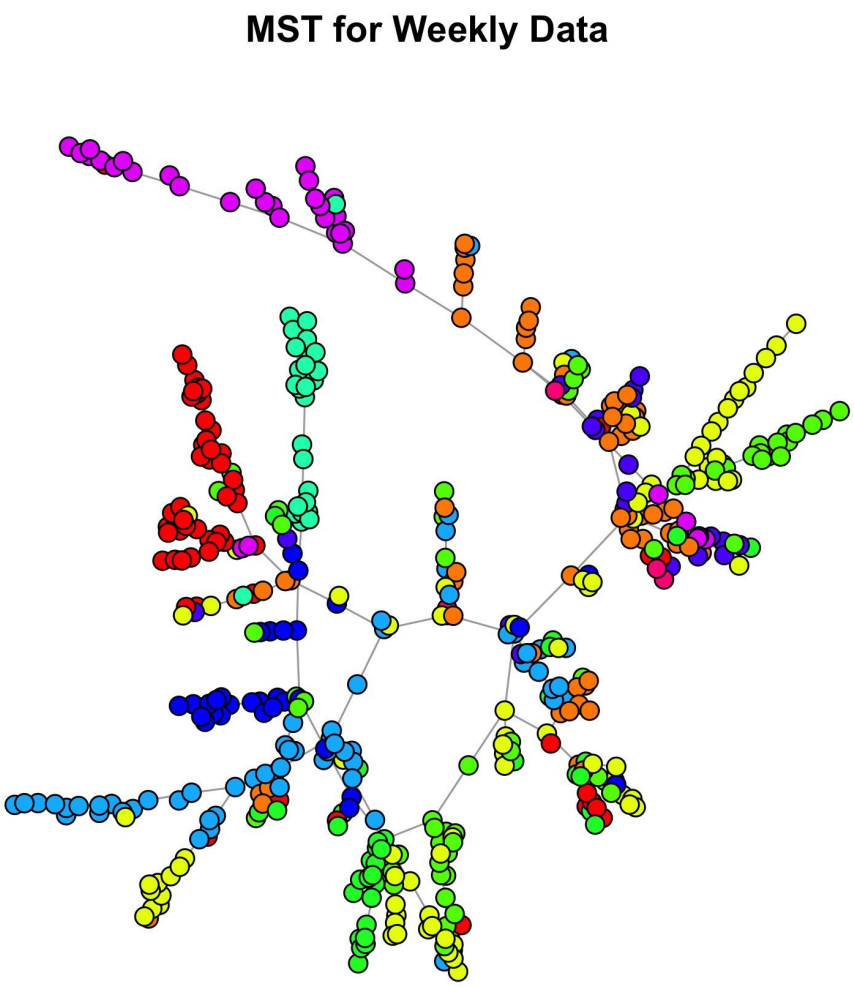
Problem 2

The distribution of the data here is unimodal with a peak at around 1.25 of over 40,000 datapoints.



Problem 3

Here color-coding was performed according to the category of a stock before using igraph's in-built mst function. We observe a notable vine distribution with stocks in related sectors being heavily correlated with each other. There are some notable stocks that serve as hubs in this distribution which are larger-cap companies that individually affect large segments of the overall economy. This plot seems to confirm the common stock market adage the sectors move together, and thus, have a high degree of correlation between them.



Problem 4

Here $\alpha$is defined as:

$$\alpha = \frac{1}{|V|} \sum_{v_i \in V} P(v_i \in S_i)$$

where S_i is the sector of a given node *i.* We can also define *P* as:

$$P(v_i \in S_i) = \frac{|Q_i|}{|N_i|}$$

Where Q_i are nodes adjacent to N that share the same sector, N_i being all neighbors to node *i*. An alternative method is to compute *P* as the following:

$$P(v_i \in S_i) = \frac{|S_i|}{|V|}$$

and S_i is the sector of node i and V is the total number of vertices.

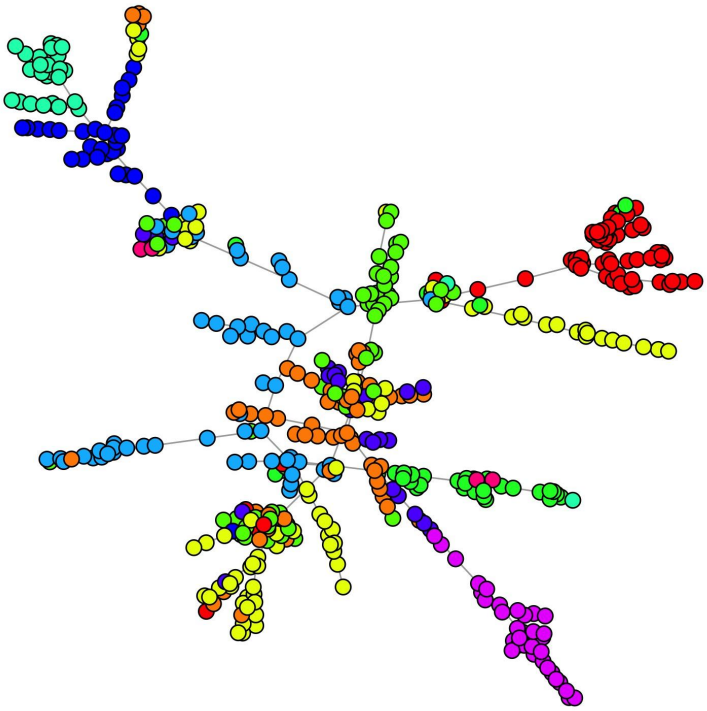
$$\alpha_1 \textbf{ - 0.829}$$

$$\alpha_2 \textbf{ - 0.114}$$

Here we report the values respectively for the two methods. The most notable fact here is the huge discrepancy between the two, one computing a relatively large value for sector determination whereas the other is the opposite. This differential is not surprising given the fact that method 1 takes into consideration the MST of the overall data graph. Given that the stocks form these elongated Vine networks, taking into account nodes further away from a node's immediate neighbors, is favorable as these clusters form nonlocal interactions. The first method only considers a node's immediate neighbors and thus loses information about this particular graph layout.

Problem 5

MST for Daily Data



In order to find weekly correlations, we only used values from a single day of the week. Otherwise the computation was identical to Q3. This distribution seems to have less clear “vines” where some sectors are not in the same segment but spill over into adjacent vines. There is, however, still a distinct clustering between sectors here.

Let’s Help Santa!

Problem 6:

Nodes	Edges
2,649	1,004,955

Problem 7:

Above are some of the nodes in G overlaid with satellite images of the corresponding geographic locations. We can see that these nodes roughly line up with intersections or large roadways near each other. This implies that the MST algorithm is reasonable in regard to recreating the connectivity of actual roadways.

Problem 8:

Here we find that ***92.2%*** of triangles satisfy the triangle inequality. This was computed by comparing the length of any two sides of a triangle to the third and final side. These lengths were the weights of the computed travel time between two points.

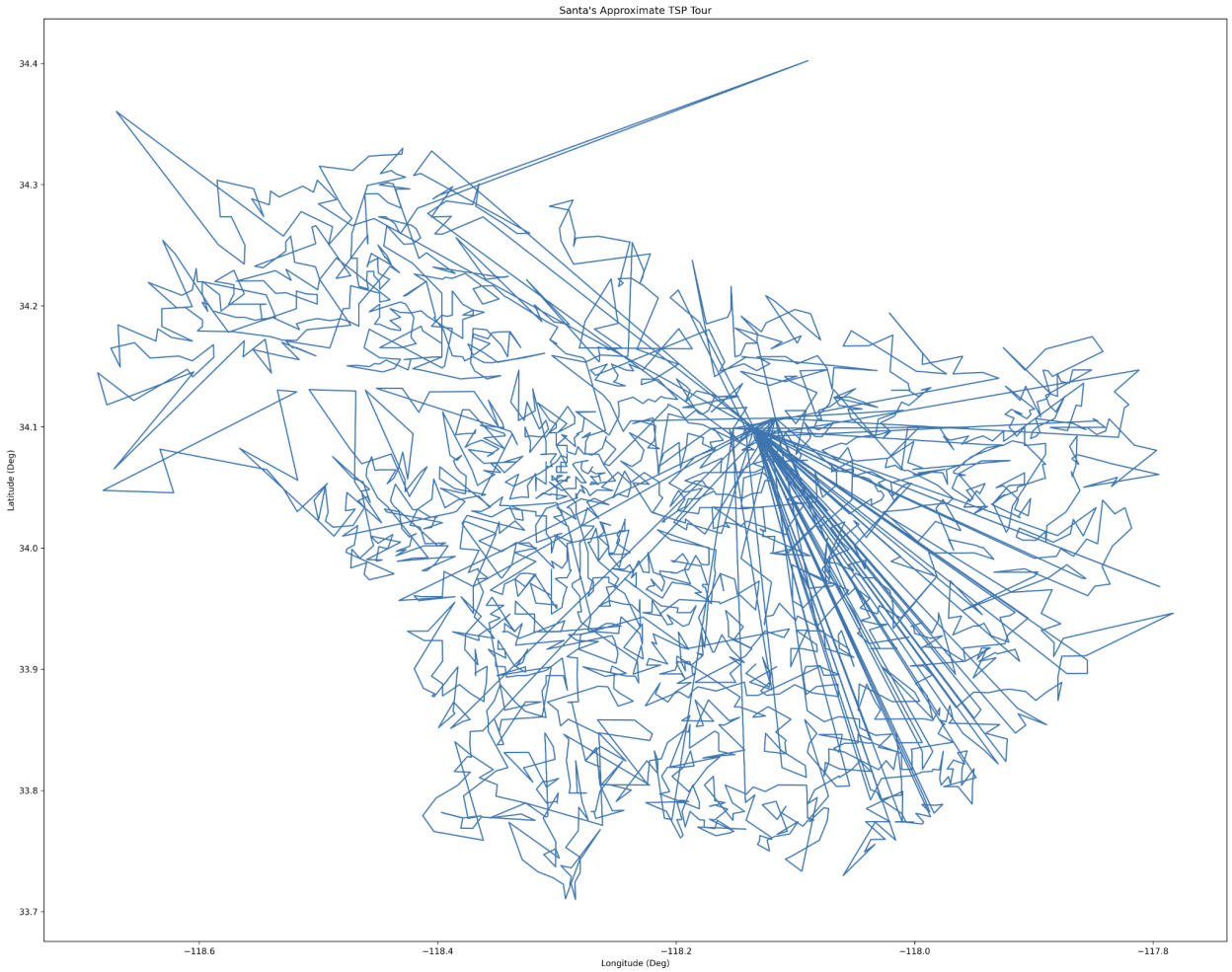
Problem 9:

Path weight(Our implementation):	**466602**
ρ	**1.73**

To compute the optimal TSP cost we computed the Euler tour of the graph, we compared this to the weight of the MST.

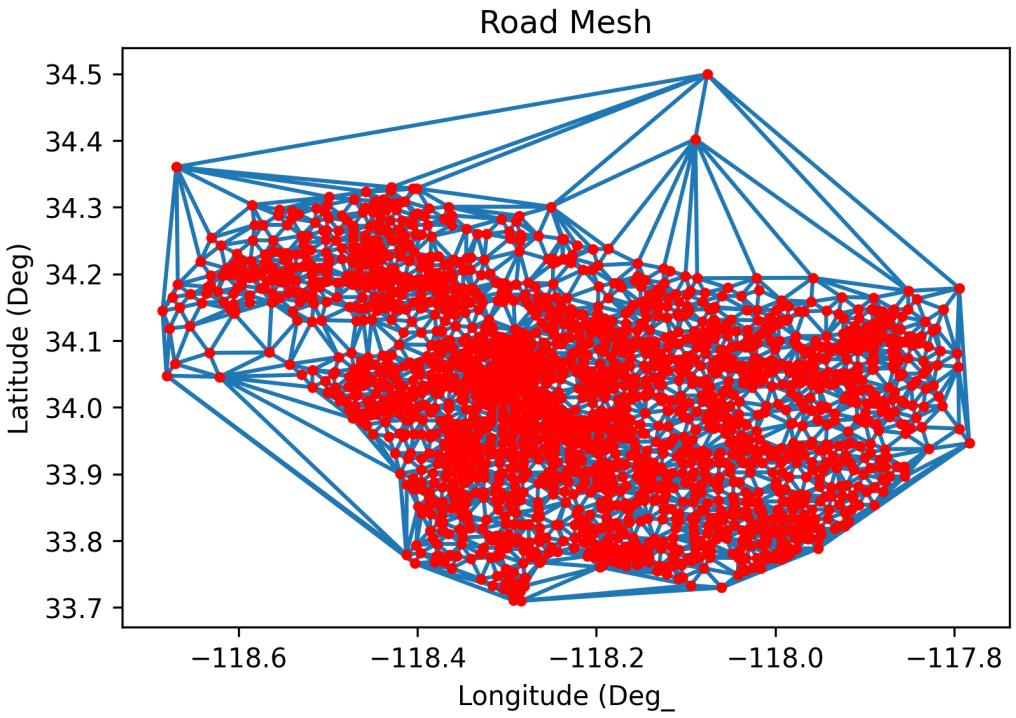
Problem 10:

Here we overlay the TSP route Santa takes with the map of LA:




Santa Clarita
San Gabriel
Mountains National
Monument
Simi Valley
Malibu
Ontario
Chino
Yorba Linda
Anaheim
Huntington
Beach
Irvine
© Mapbox © OpenStreetMap

Question 11:


Road Mesh
34.5
34.4
34.3
34.2
34.1
34.0
33.9
33.8
33.7
Latitude (Deg)
−118.6
−118.4
−118.2
−118.0
−117.8
Longitude (Deg_

Delauney triangulation operates on a simple rule that constructs triangulations where all points in a set exist on the edge of a circumcircle of these triangulations. The above is the results of Delaunay triangulation of the previous dataset. Here the result is largely as expected given the rule of triangulation based on circumcircles. This means that LA Bay, for example, will be mischaracterized as containing roads because it is convex and the triangles of its resulting circumscribing circles jut into the bay as a result. We also see a greater deal of interconnectivity in the denser parts of the city, largely due to the enforcement of the rule that no point lies within the circumcircle. This results in much smaller circumcircles and thus shorter, denser approximated road segments in the denser parts of the city. Furthermore, this approximation technique will overpopulate the connections to more "rural" parts of the city to the north. In reality there are fewer connections to these further off points, also to be expected when this methodology does not consider topology or population density.

Question 12:

Here we want to compute the differential distance of each car. This is computed using the following (in units of km/hr for velocity):

$$dx = 0.005\ km + v\,(2\ s\ \frac{1\ hr}{3600\ s}) \rightarrow \text{distance/car}$$

From here we can rewrite velocity to yield the differential of time:

$$v = \frac{dx}{dt} \rightarrow dt = \frac{dx}{v} = \frac{0.005}{v} + \frac{1}{1{,}800}\ \text{time/car}$$

Ultimately we want units of cars/time so we invert the above and double it to account for the two lanes of traffic on the road:

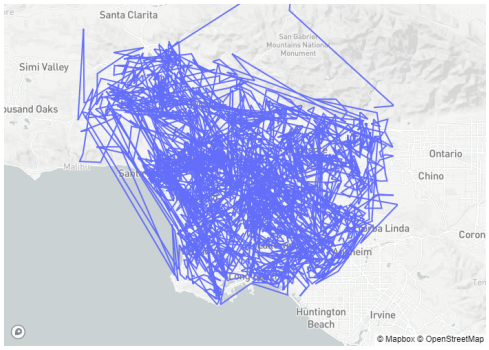
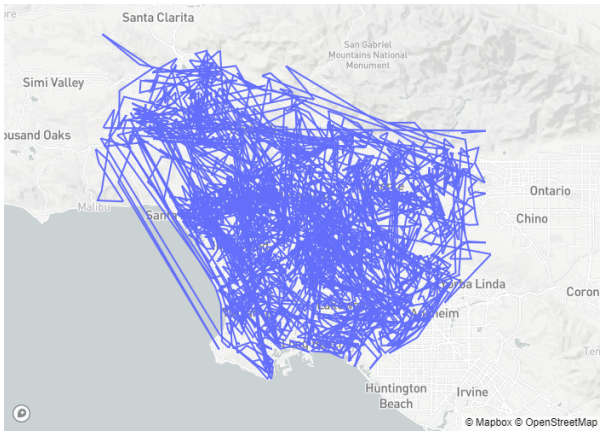
$$\text{Capacity} = \frac{cars}{time} = \frac{1}{\frac{0.005}{2v} + \frac{1}{3600}}$$

Question 13:

Using the formula above we calculated the flow for each road and further used this information to obtain the maximum number of cars that can commute between Malibu and Long Beach. The number of cars in each direction was **11074** per hour and the number of disjoint paths was **4**.

Question 14:

Threshold	***Graph***

No threshold	
3σ threshold (from mean of the traveling time); Threshold = 626 s	
Threshold = 100 s	

Graph generated by the means of triangulation has some non-physical edges going through the bay or through the mountains. Too remove them we applied different thresholds. One threshold was based on the 3σ rule which helped to remove the edges from the mountain area. Total number of edges for this case decreased from 7758 (for the no threshold case) to 7638. The threshold value was equal to ~626 s.

To remove the non-physical edges from the bay we further decreased the threshold and were able to only remove all of those edges for Threshold as low as 100 s. Total number of edges in this case got down to 1286. These results tell us that the thresholding method works partially (i.e. helped for the edges at the mountain, but works worse for the edges at the bay).

Question 15:
For the case of the 3σ based threshold. The maximum number of commuting cars per hour between Malibu and Long Beach stayed the same at **11074** and the number of disjoint paths remained **4**. The deleted edges are mainly in the middle of the graph and not near the source and destination as seen in the figure, allowing the number of disjoint paths to remain the same. Since these 4 paths are likely the bottleneck for the traffic flow, the overall traffic flow also remains the same.

Q16: Define your own Task

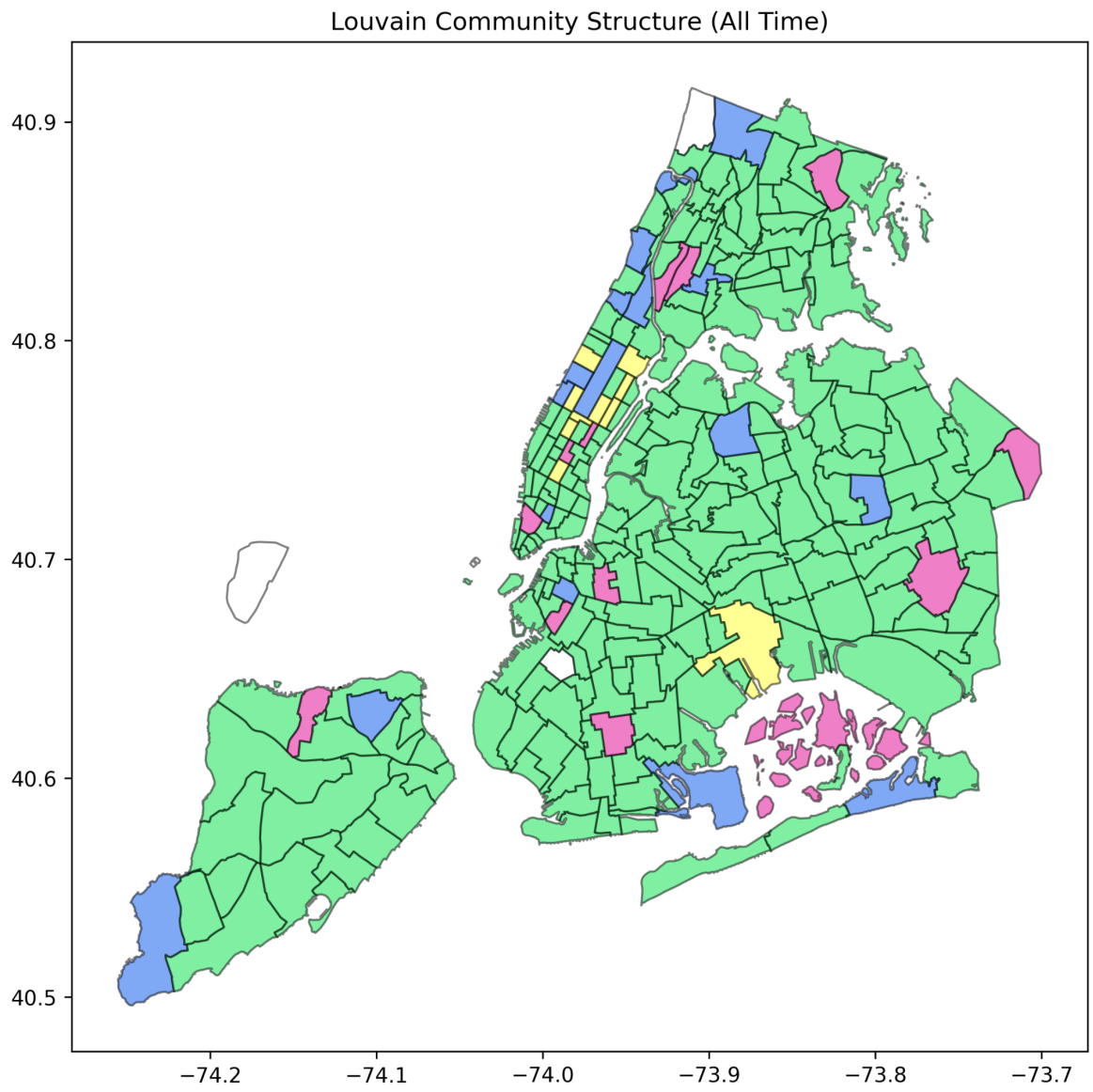
We seek to analyze rideshare trips in NYC and look at the community structure and common trends.

Our dataset contains every rideshare trip in NYC during January 2020. For our analysis we focus on 4 data fields: start time, end time, pick up zone, drop off zone. The granularity of the pick up and drop off locations limit our analysis in some respects but provide a convenient way to segment areas of the city and look at how people move between different areas.

We create an undirected graph with edge weights representing the number of rides between nodes. Each node represents a zone as mentioned above. We seek to determine which areas of the city are most closely “connected” and do so by applying the Louvain method for community detection. The algorithm optimizes for modularity of the graph, in this case weighted modularity since the graph is likely to be strongly connected and thus unweighted modularity would provide little insight. We have the following as the weighted modularity that we seek to maximize:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

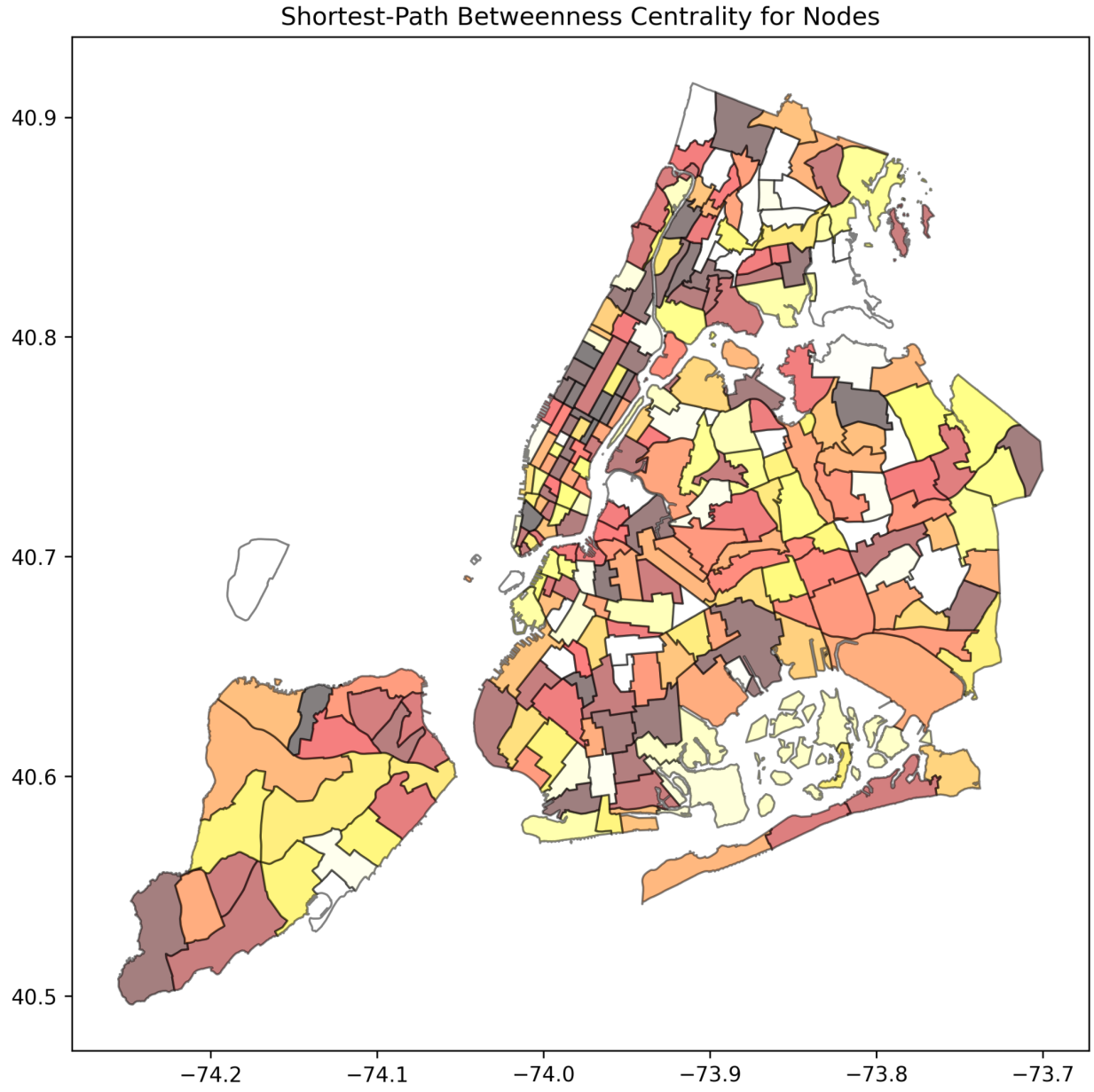
We then assign a color to each discovered community and plot a map of the city:



Next, we look at how the structure changes throughout the day by again using the Louvain algorithm, except in increments of an hour and plot the changes. The edge thickness represents the number of trips and due to the highly uneven distribution of trips, we see that most edges between residential areas are not at all visible. Furthermore, we plot the degree distribution and edge weight distribution at each hour. An animation of these plots can be found here: https://i.imgur.com/7RQJZTn.gif

We also look at the betweenness centrality for each node. The betweenness centrality of a node is calculated by first calculating the shortest path for every pair of nodes (zones in our case) using the number of trips as weights. The centrality is then the number of these paths that pass through a given node, hence serving as a measure of how ‘central’ the node is to the network.

We invert the weights as a high weight (corresponding to a large number of trips) should be at the ‘center’ of our network. Darker colors indicate a higher centrality value.



Next, we recreate the graph as a directed graph with an edge from source -> destination again with weight equal to number of trips. This allows us to analyze the net flow for each node and see how people move throughout a given day. We again plot over 24 hours and show the net flow. Darker colors represent a higher outflow and lighter colors represent a smaller or negative flow. We take the log of the net flow to represent the color as otherwise it is difficult to differentiate the differences. We also plot the degree distribution (both in/out) for each hour:

https://i.imgur.com/XpFaj8c.gif

Lastly, we look at the probability density function for trip length with a couple extremely long rides removed. It is interesting to note that one ride lasted ~23.9 hours according to the dataset so perhaps they were attempting to set a record. In any event we see that most trips are no longer than 30 minutes.