

ECE 232E: Large-Scale Social and Complex Networks: Models and Algorithms

Project 5: Graph Algorithms

Akshay Sharma (504946035)

Anoosha Sagar (605028604)

Nikhil Thakur(804946345)

Rahul Dhavalikar (205024839)

1. Stock Market

1.1. Return correlation

In this part of the project, we will compute the correlation among log - normalized stock-return time series data.

- $p_i(t)$ is the closing price of stock 'i' at the t^{th} day.
- $q_i(t)$ is the return of stock 'i' over a period of $[t - 1, t]$.

$$q_i(t) = \frac{p_i(t) - p_i(t - 1)}{p_i(t - 1)}$$

- $r_i(t)$ is the log-normalized return stock 'i' over a period of $[t - 1, t]$.

$$r_i(t) = \log(1 + q_i(t))$$

The correlation between the log-normalized stock-return time series data of stocks 'i' and 'j' is defined as below,

$$\rho_{ij} = \frac{\langle r_i(t)r_j(t) \rangle - \langle r_i(t) \rangle \langle r_j(t) \rangle}{\sqrt{(\langle r_i(t)^2 \rangle - \langle r_i(t) \rangle^2)(\langle r_j(t)^2 \rangle - \langle r_j(t) \rangle^2)}}$$

Where, $\langle \cdot \rangle$ is a temporal average on the investigated time regime (for our data set it is over 3 years).

- 1. Provide an upper and lower bound on ρ_{ij} . Also, provide a justification for using log-normalized return ($r_i(t)$) instead of regular return ($q_i(t)$).**

Ans: ρ_{ij} indicates the correlation between the stock 'i' and stock 'j'. A range of correlation ranges from +1 to -1. The relationship is positive when the two variables move in the same direction (both up or both down) and is negative when they move in opposite direction (one up, other down). The value of

zero reflects no correlation at all. A value of +1 shows perfect positive correlation while -1 is perfect negative correlation. Example, If both stocks move up by 5%, then they can be said to have a perfect positive correlation.

The upper and lower bound of ρ_{ij} can be derived as follows:

From the definition of correlation above, we can see that both the variances (the denominators) are positive.

If two variables are perfectly uncorrelated, then their covariance (numerator) is 0.

$$Cov(X, Y) = E[(X - \bar{X})(Y - \bar{Y})] = E[XY] - E[X]E[Y]$$

In case of correlation, $E[XY] = E[X]E[Y]$, hence $Cov(X, Y) = 0$

Now, on applying the Cauchy-Schwarz inequality

$$|Cov(X, Y)|^2 \leq Var(X)Var(Y)$$

$$\therefore |Cov(X, Y)| \leq \sqrt{Var(X)Var(Y)}$$

Putting this result in the definition of correlation, we get

$$|\rho| = \left| \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}} \right| \leq \frac{\sqrt{Var(X)Var(Y)}}{\sqrt{Var(X)Var(Y)}} = 1$$

From the above result we can say that $|\rho| \leq 1$ or $-1 \leq \rho \leq 1$

There are a few reasons for using the log-normalized return instead of regular return.

- Logarithm have the important property of additivity which makes them easier to use in various algorithms.
- When returns are very small, log normalization ensures that they are close in value to raw returns
- Log normalization adds numerical stability - multiplying small numbers is subject to arithmetic underflow and is a serious problem for many applications. Hence we can modify it to a summation via logs to ensure stability.

1.2. Constructing correlation graphs

Now, we construct a correlation graph using the correlation coefficient computed in the previous section. The correlation graph has the stocks as the nodes and the edge weights are given by the following expression.

$$w_{ij} = \sqrt{2(1 - \rho_{ij})}$$

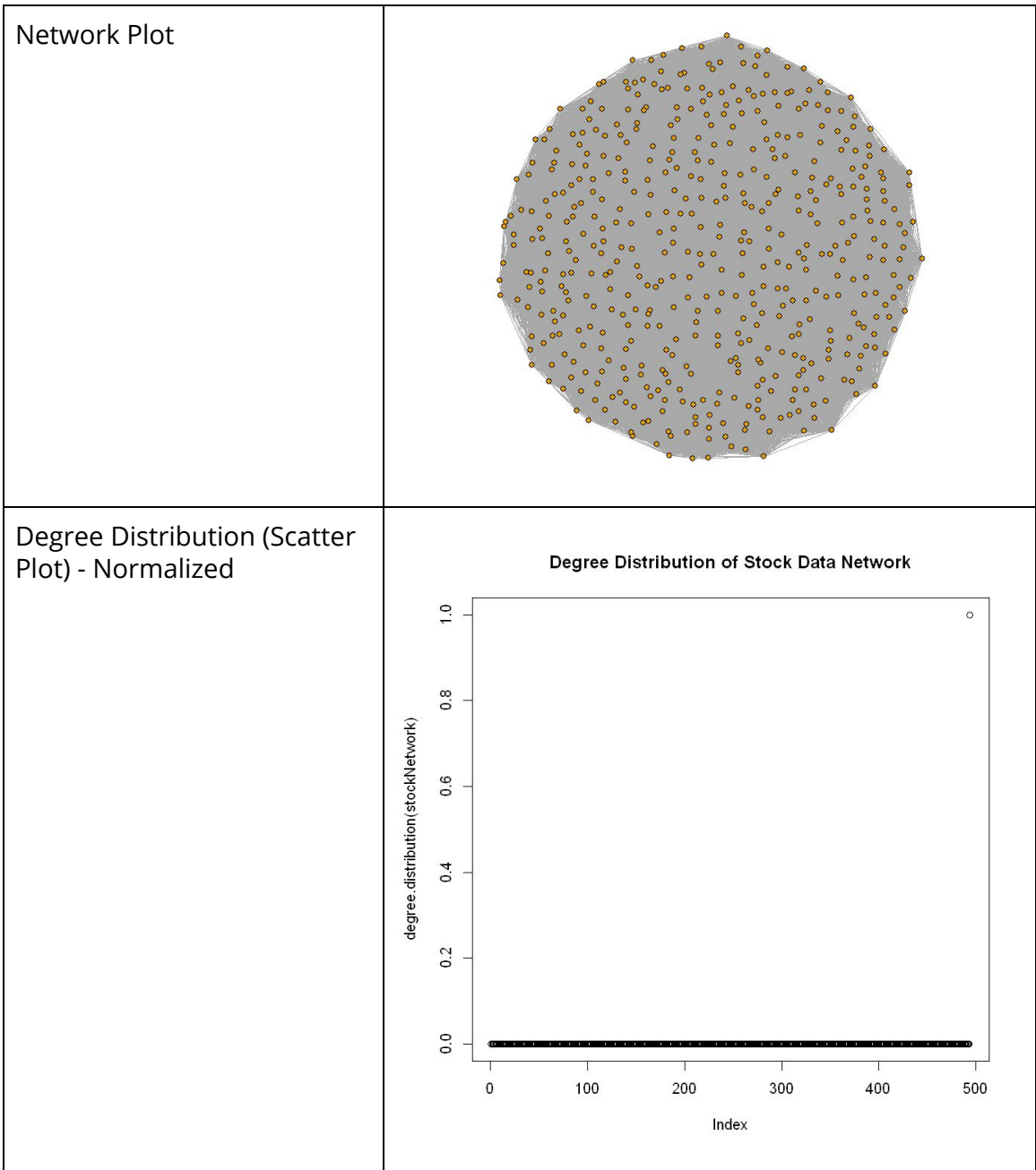
In this part of the project, we will extract the MST of the correlation graph and interpret it.

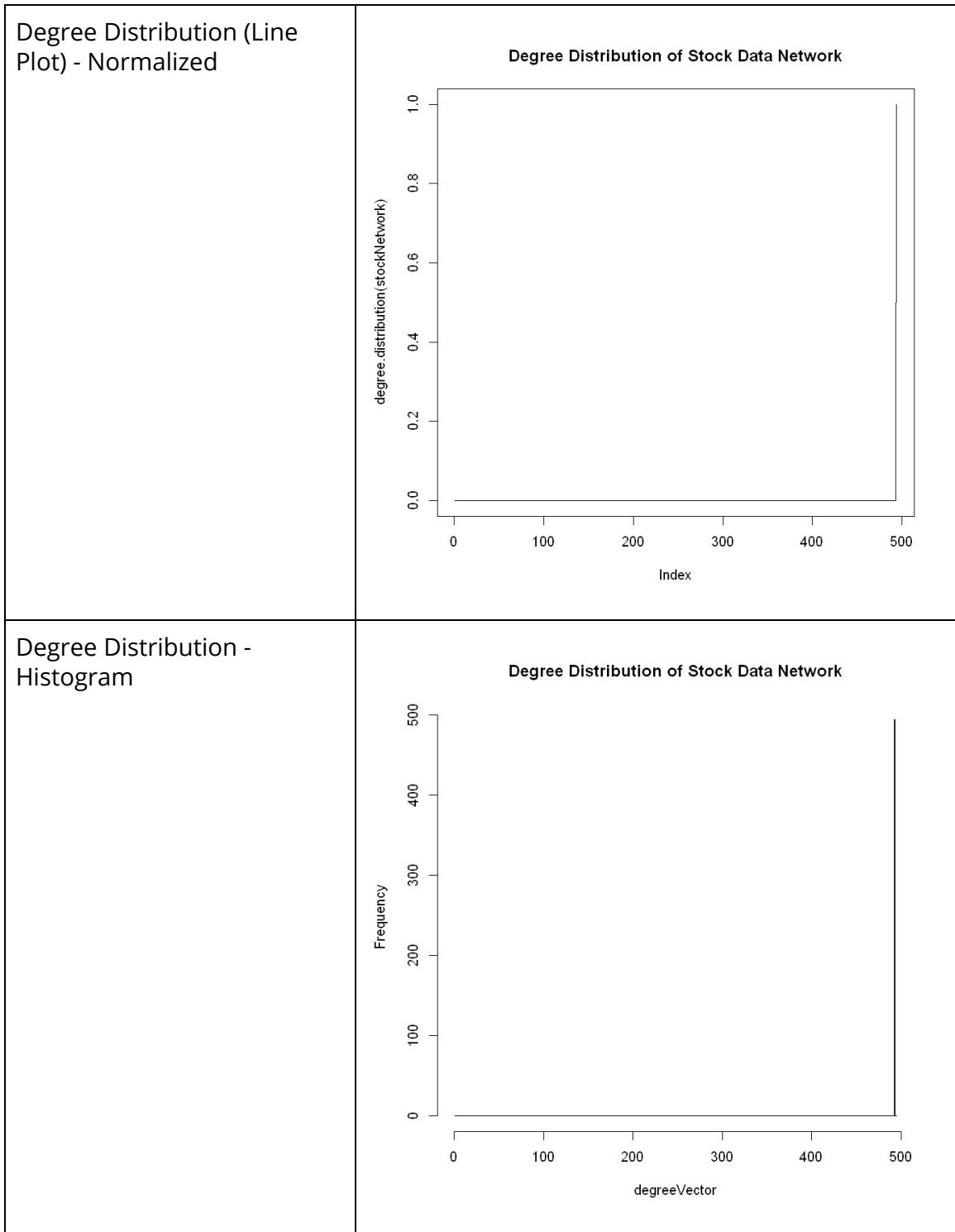
2. Plot the degree distribution of the correlation graph and a histogram showing the un-normalized distribution of edge weights.

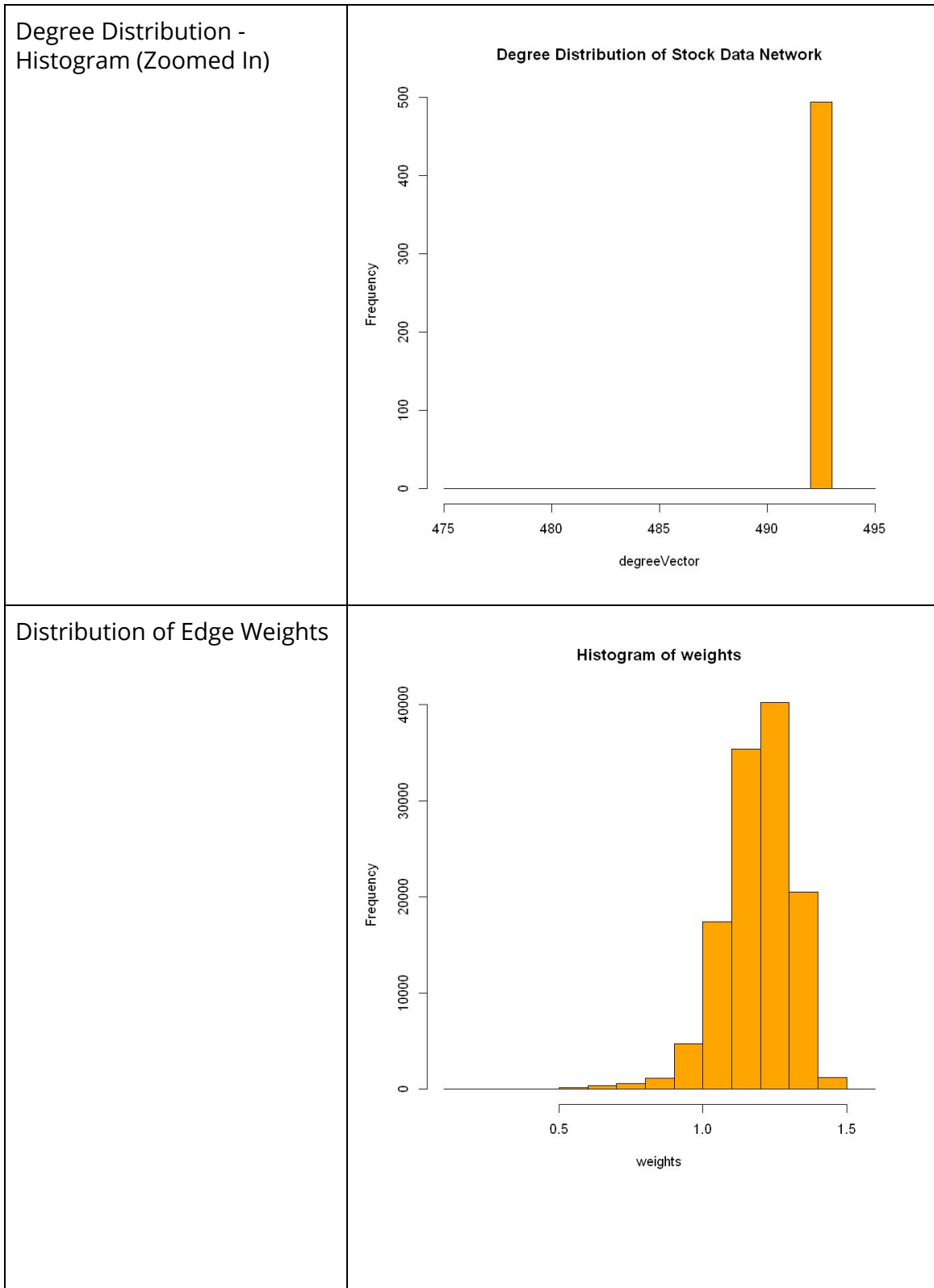
Ans: We first trim our dataset to contain stocks which have the complete data for the three years. Hence we now contain a network from 494 stocks. As this network is a cross correlation matrix, we expect to have $n(n - 1) / 2$ edges which is what we observe.

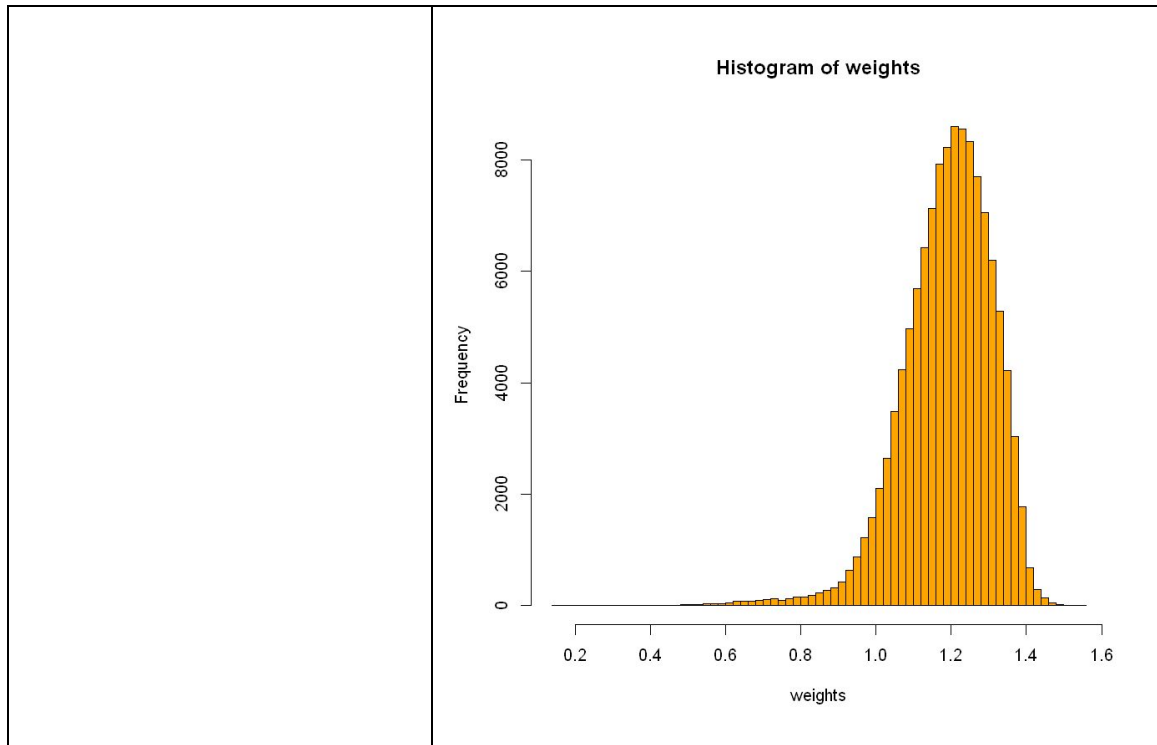
The details of the network and the corresponding degree and weight distributions are shown below:

Number of Vertices	494
Number of Edges	121771
Diameter	1.54827216684821









From the graphs of the degree distribution, we can observe that every vertex in the network has the same degree that is 493 which indicates that it is a fully connected network.

On inspecting the edge weight distribution, we observe that it mimics the log-normal distribution. An interesting thing to note is that for our data, applying log normalization does not result in a major change as compared to when don't apply this normalization. This is due to the fact that as our values of $q(t)$ {return of stock over a period of $[t-1, t]$ } are very small, log normalization retains these values and hence we obtain similar edge weights in both cases of using normalization and not.

1.3. Minimum Spanning Tree (MST)

3. Extract the MST of the correlation graph. Each stock can be categorized into a sector, which can be found in Name_sector.csv file. Plot the MST and color-code the nodes based on sectors. Do you see any pattern in the MST? The structures that you find in MST are called Vine clusters. Provide a detailed explanation about the pattern you observe.

Ans: A minimum spanning tree (MST) is a subset of the edges of a connected, edge-weighted graph which connects all the vertices together, without any cycles and with the minimum possible edge weight. In this question, we find the MST of the network created in the previous question, i.e., stock market data. For this purpose we use the `minimum.spanning.tree` algorithm present in R which makes use of Prim's algorithm for finding the MST.

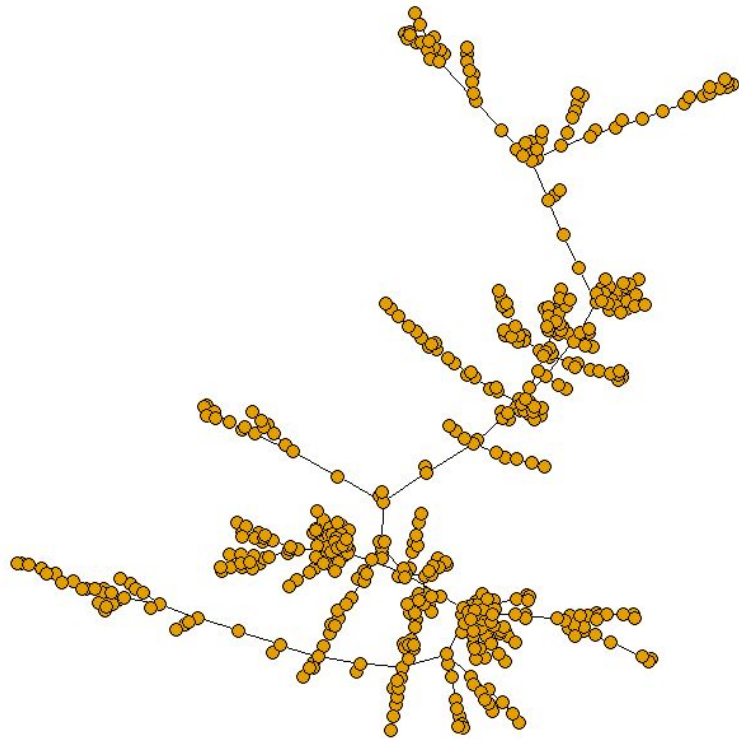
Prim's algorithm can be described as follows:

1. Initialize a tree with a single vertex, chosen randomly from the graph
2. Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree
3. Repeat step 2 (until all vertices are in the tree)

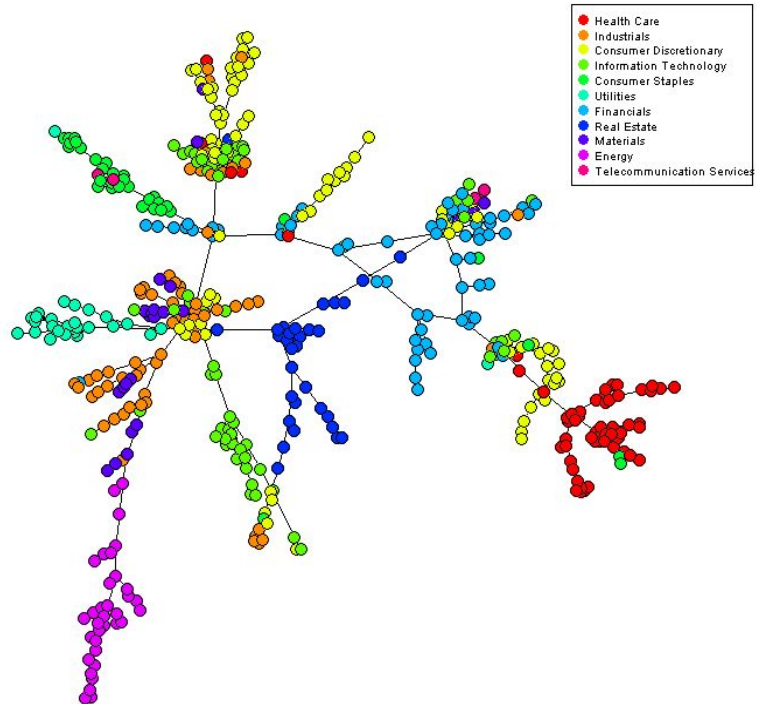
The major advantage of using Prim's algorithm is that it is significantly faster in the case of a dense graph. On computing the MST on our network, we get a tree with the following statistics:

Number of Edges	493
Diameter	27.3576195646759

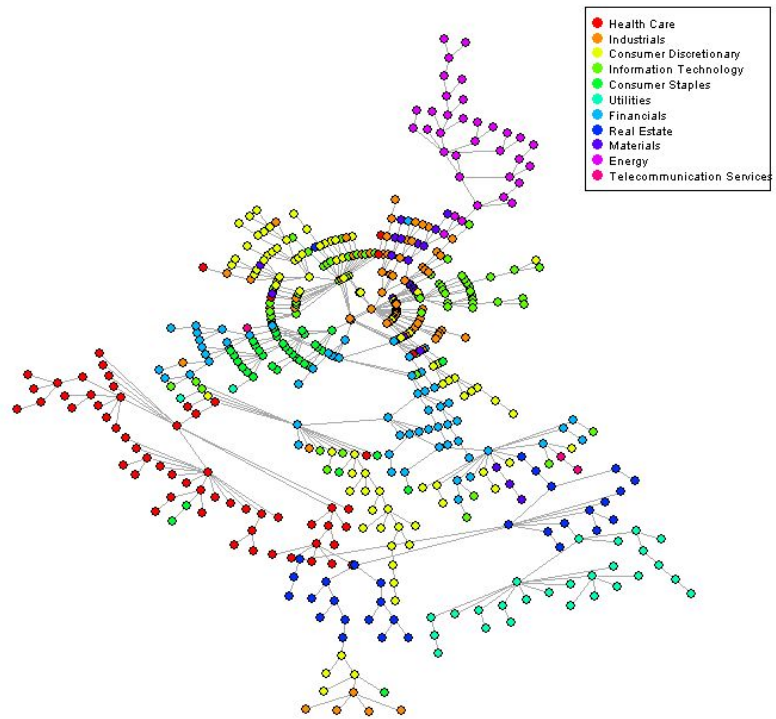
Plot of MST



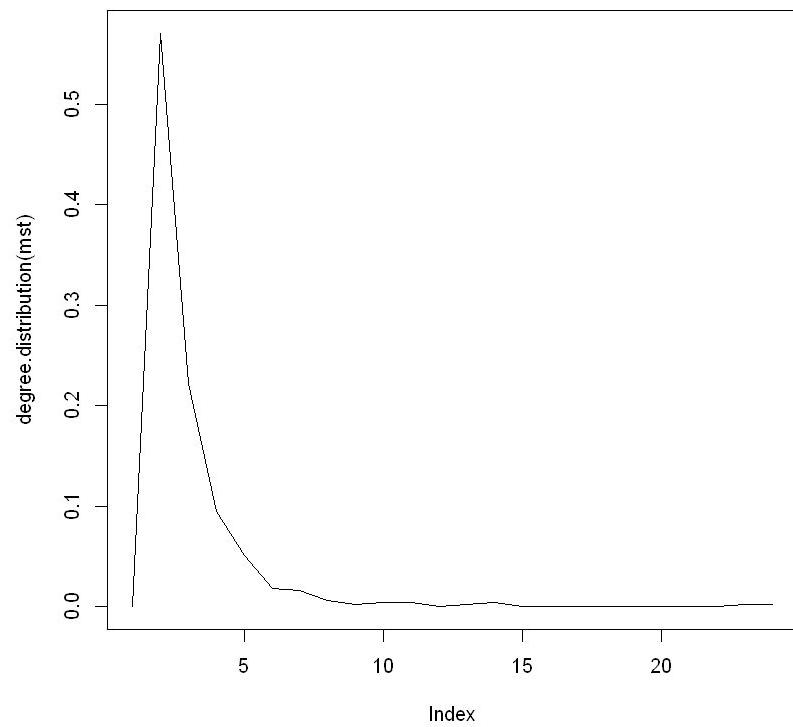
Color Coded Plot of MST



Using Reingold Tilford
Layout



Degree distribution of
MST



We observe that the MST reveals a Vine like structure consisting of different clusters. On further inspecting the MST clusters using the Reingold Tilford layout, we see that the stocks represented by each cluster's nodes turn out to belong to the same economic sector.

The intuition behind MST showing the information of economic sectors is that the investors tend to have the same investing strategy for stocks in the same economic sector in the long run, which produces a larger cross correlation between them on larger time scales. While constructing the MST, when the algorithm reaches a stock of a particular type, the connected edges of this stock with the smallest weights are from stocks of the same type due to their high values of correlation. This is the reason why stock of similar type form a Vine like structure.

1.4. Sector clustering in MST's

In this part, we want to predict the market sector of an unknown stock. We will explore two methods for performing the task. In order to evaluate the performance of the methods we define the following metric.

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

where S_i is the sector of node 'i' and

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

where Q_i is the set of neighbors of node 'i' that belong to the same sector as node 'i' and N_i is the set of neighbors of node 'i'. Compare α with the case where,

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

4. Report the value of α for the above two cases and provide an interpretation for the difference

Ans: The second alpha represents a more global metric, as it contains the total number of vertices in the denominator as opposed the first alpha which contains only the neighbours in the denominator and can be considered as a metric which has been defined locally. The first alpha is specific to the edges in the MST whereas the second one includes all the edges in the entire graph.

The value of the first alpha is higher than the second alpha because as we observed earlier in the structure of the MST, we see distinct vines with stocks of a sector mostly connected with other stocks of the same sector. Due to this, the set Q_i will contain a large number of stocks which will result in a larger value of alpha in this case.

For the stock network as a whole, as it is a fully connected network, the factor $|S_i|/|V|$ will be lower which will result in a smaller value of alpha as we have observed in our findings.

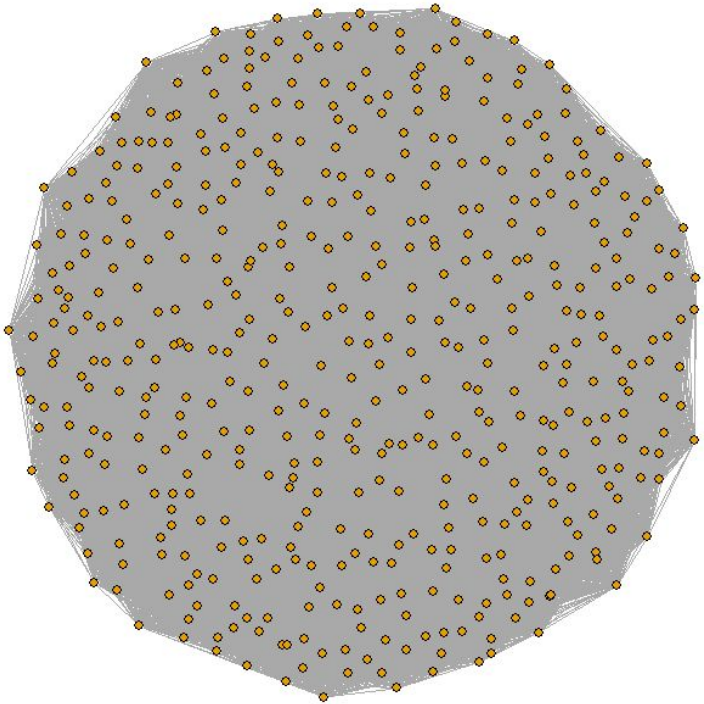
First Case α	0.8289300775306759
Second Case α	0.11418807061253222

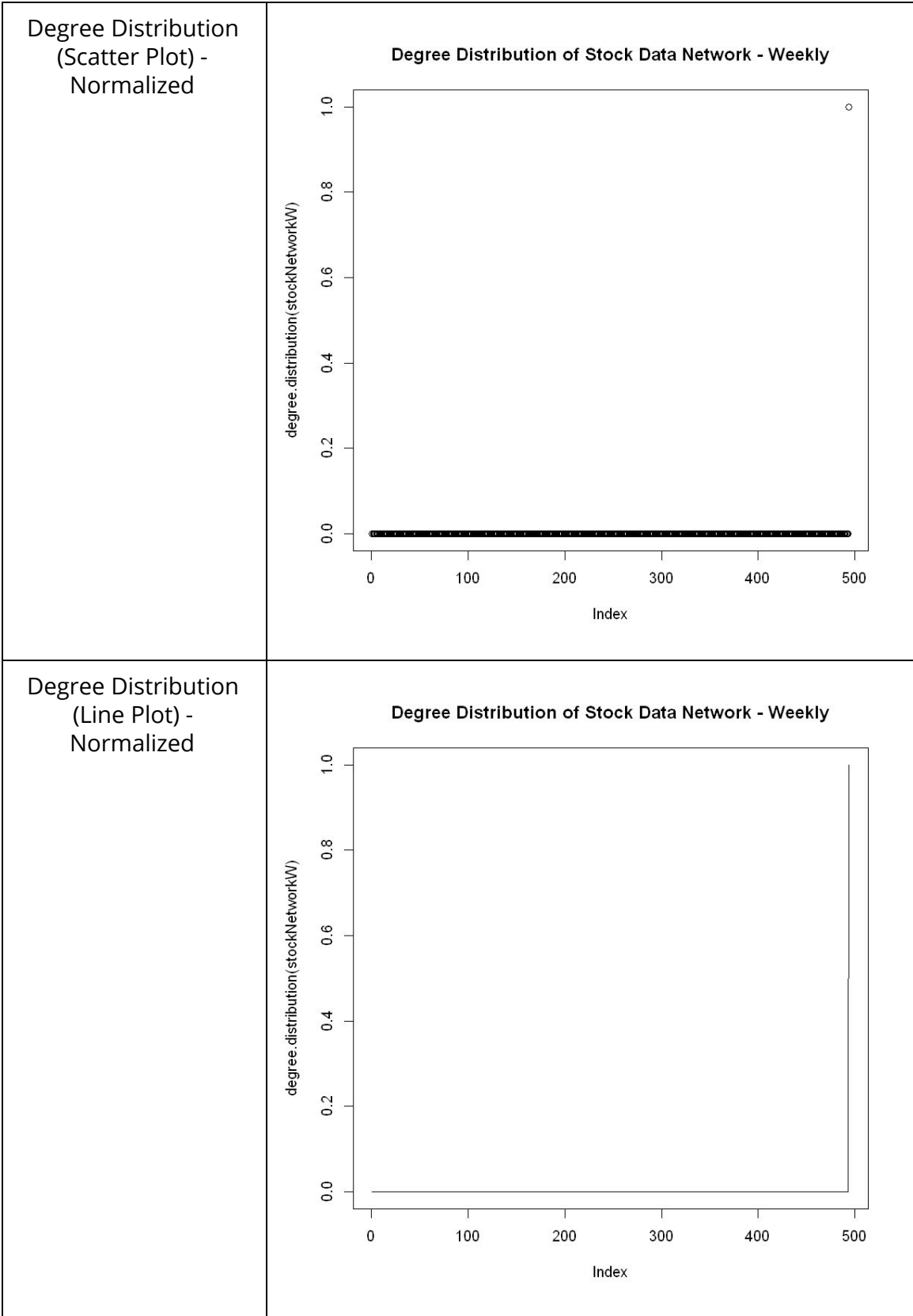
1.5. Correlation graphs for weekly data.

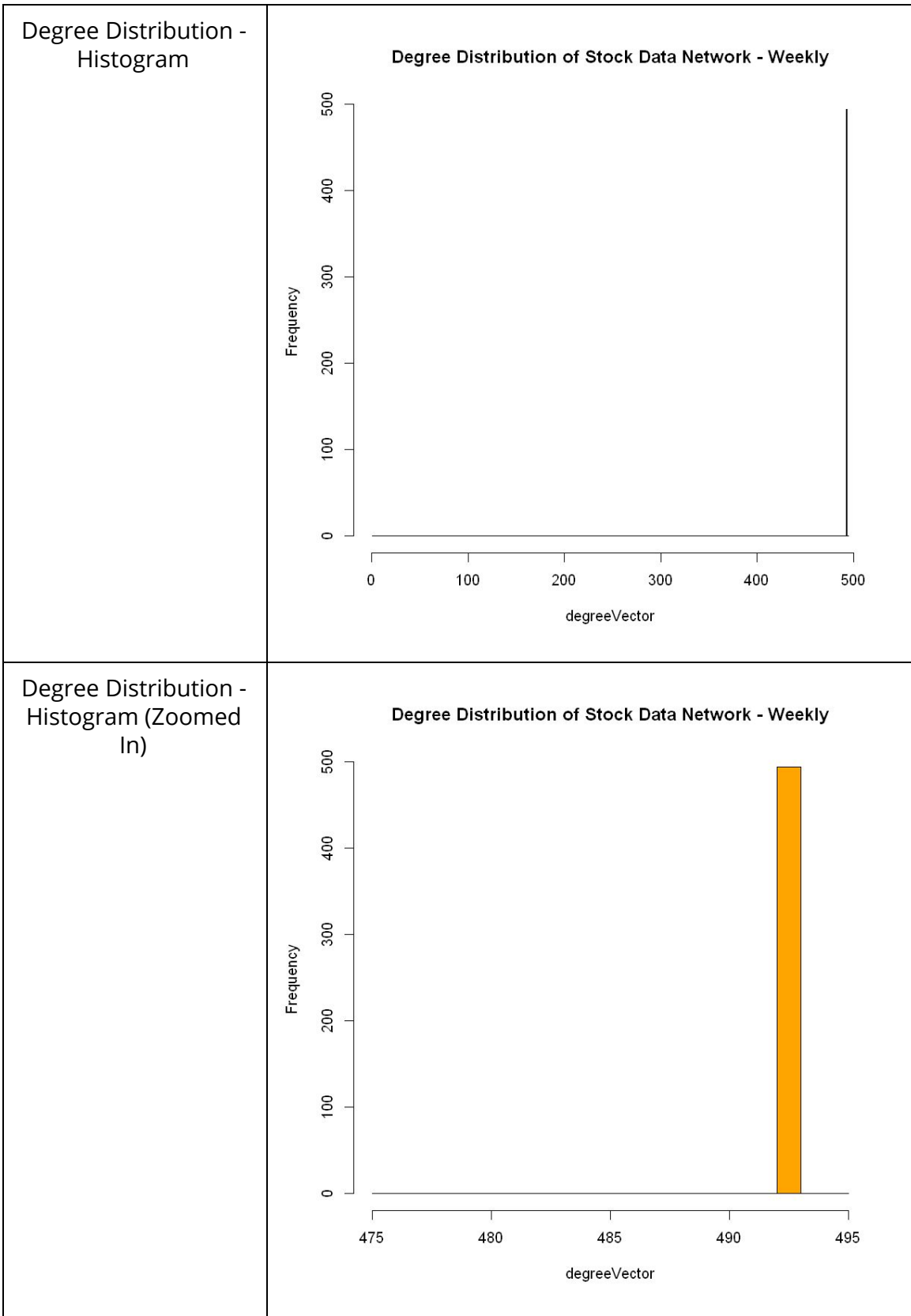
In the previous parts, we constructed the correlation graph based on daily data. In this part of the project, we will construct a correlation graph based on weekly data. To create the graph, sample the stock data weekly on Mondays and then calculate p_{ij} using the sampled data. If there is a holiday on a Monday, we ignore that week. Create the correlation graph based on weekly data.

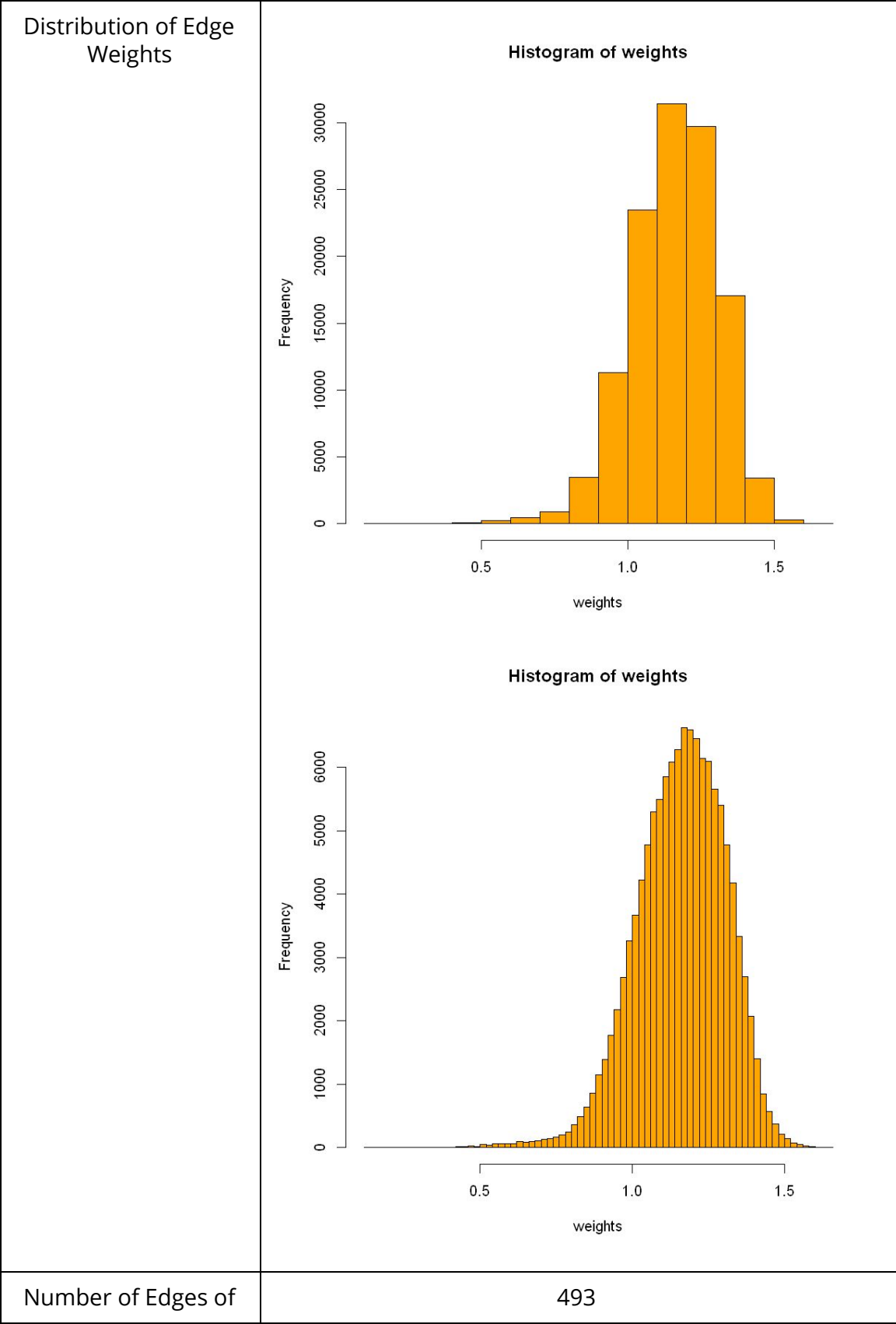
5. Extract the MST from the correlation graph based on weekly data. Compare the pattern of this MST with the pattern of the MST found in question 3.

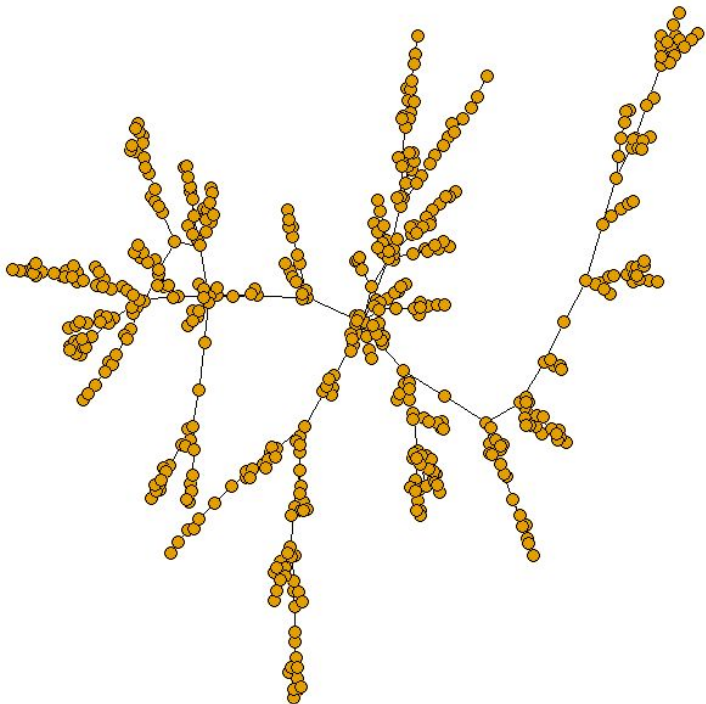
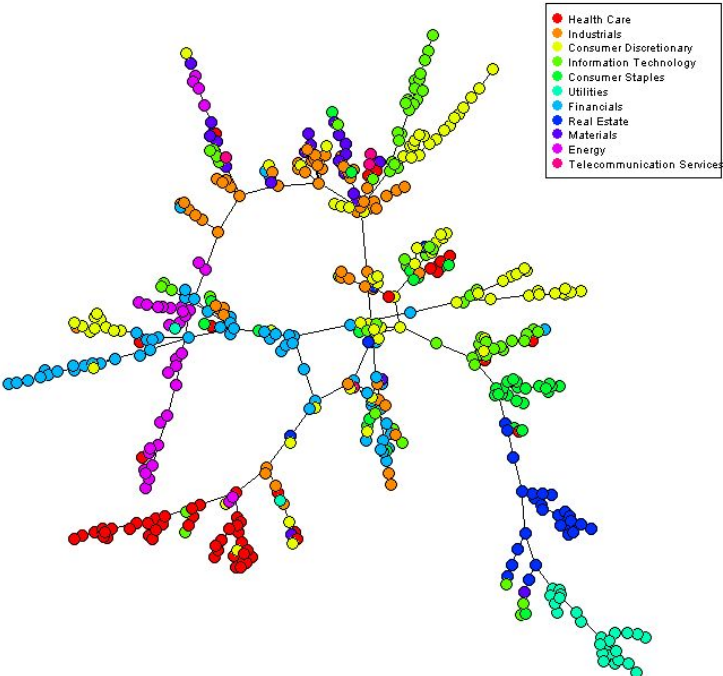
Number of Vertices	494
--------------------	-----

Number of Edges	121771
Diameter	1.65421867390361
Network Plot	

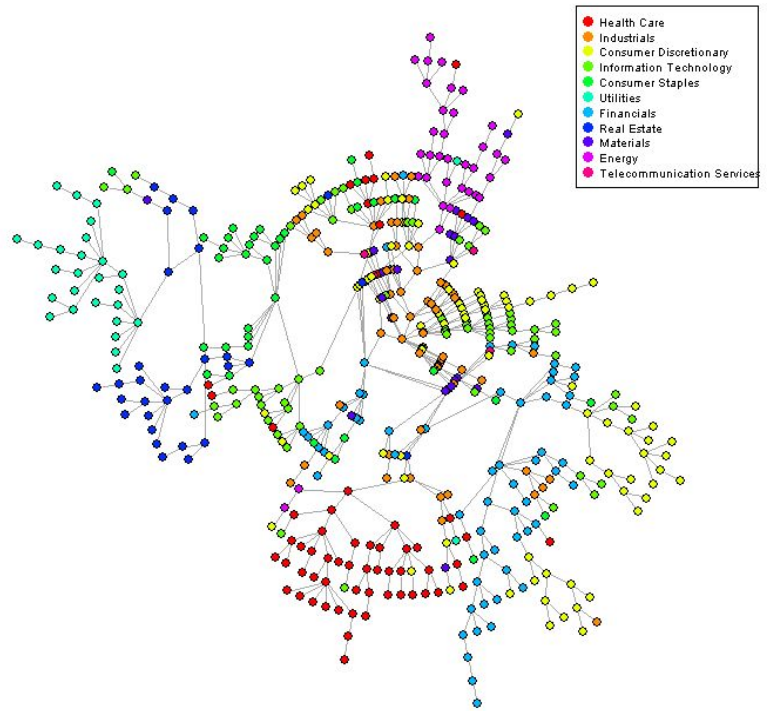




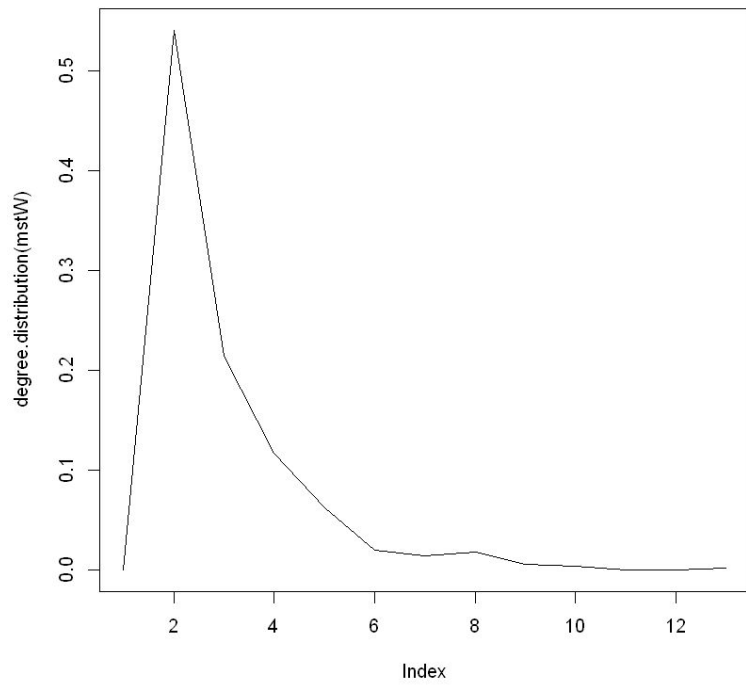


MST	
Diameter of MST	20.2774848297316
Plot of MST	
Color Coded Plot of MST	

MST With Reingold
Tilford Layout



Degree Distribution of
MST



First Case α	0.7429696034959192
Second Case α	0.11418807061253222

In this part, we construct a new network by only sampling weekly data. There are some minor differences observed in the network as we have reported above. We do not observe any major difference between the MST constructed in this part to the daily network's MST but we see that the first alpha value is lower than the value obtained for the daily data. This is due to the fact that the daily data is more fine grained and captures richer information than the weekly data.

2. Let's Help Santa

2.1. Report the number of nodes and edges in G.

In this part, we have made use of Uber Data for the San Francisco area for the month of December 2017. On creating the graph according to project specifications, we observe that there are 3 components in the network which are reported below. We have considered only the largest component which has a size of 1880 nodes as our graph G for future parts.

Hence, we have 1880 nodes in the network with 311802 edges as shown below:

Clusters in graph	3
Sizes of Clusters	1880, 5, 2
Number of Nodes in G	1880
Number of Edges in G	311802

2.2. Build a minimum spanning tree (MST) of graph G. Report the street addresses of the two endpoints of a few edges. Are the results intuitive?

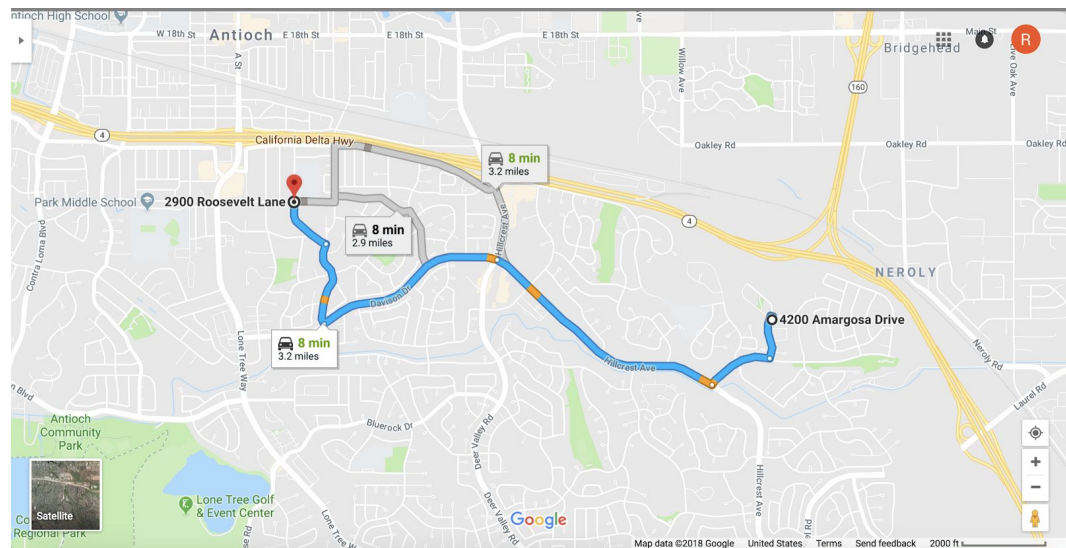
We have used the same function as used for finding the MST of stock market data.

Graph Max Weight	6439.18
Graph Min Weight	21.9
MST Max Weight	801.76
MST Min Weight	21.9

The above table shows some statistics about the MST.

Some edges

Location		Edge Weight
From	To	
1624	828	158.965
2900 Roosevelt Lane, Antioch	4200 Amargosa Drive, Antioch	



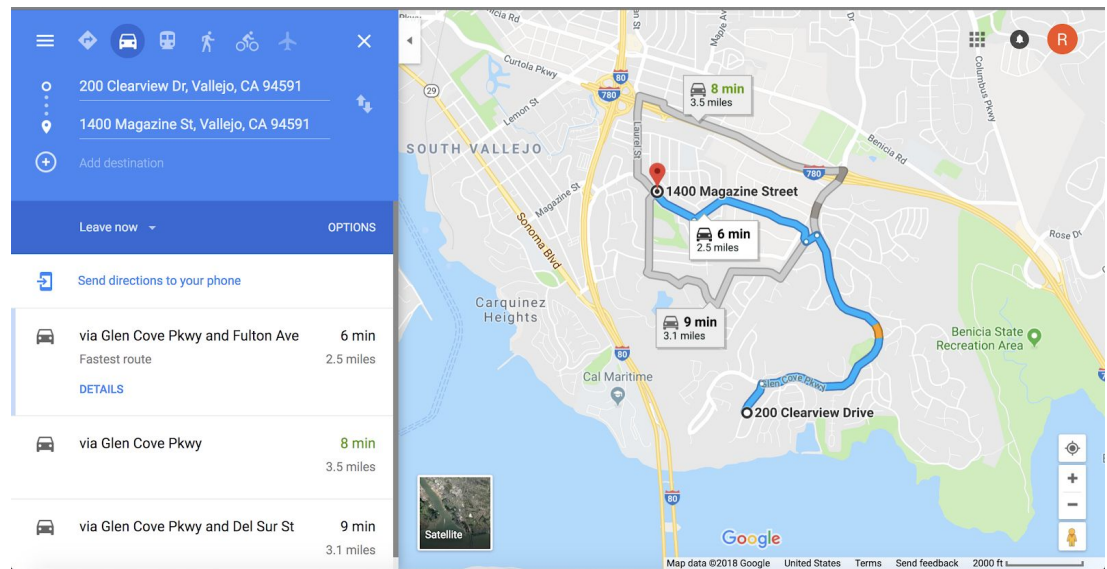
Location		Edge Weight
From	To	
1732	1450	93.535

0 Longview Drive, Westlake, Daly City	0 Brookhaven Court, Fairmont, Pacifica	
--	---	--

Location		Edge Weight
From	To	
455	721	130.305
3400 Mauricia Avenue, Santa Clara	600 Hobart Terrace, Santa Clara	

Location		Edge Weight
From	To	
1550	1484	147.265
30800 San Clemente Street, Hayward	29300 Lassen Street, Tennyson - Alquire, Hayward	

Location		Edge Weight
From	To	
2022	749	87.03
200 Clearview Drive, Vallejo	1400 Magazine Street, Vallejo	



From the statistics in table, we see that the max edge weight in original graph was 6439.18 which reduced to 801.76 in the MST.

Also as expected the edges weights of the MST are low and after plotting them on Google Maps we see that most of the edges and locations in the MST are close by and in the same area.

2.3. Determine what percentage of triangles in the graph (sets of 3 points on the map) satisfy the triangle inequality. You do not need to inspect all triangles, you can just estimate by random sampling of 1000 triangles.

In this task, we have taken combinations of 3 edges to form triangles. We get a lot of triangles and we have used 1000 random triangles.

The triangle inequality states that for any triangle, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side.

Number of Triangle Combinations	95036715
Percentage of Triangles Satisfying Triangle Inequality	93.05%

2.4. Find the empirical performance of the approximate algorithm:

The input to the traveling salesman problem (TSP) is an undirected graph with edge weights, and the optimal output is a cycle that visits every vertex exactly once (a tour) with minimum total weight. It is not possible to find such a tour in polynomial time unless $P=NP$. However, for edge distances that satisfy the triangle inequality $d(x,y) \leq d(x,z) + d(z,y)$ for all x, y, z , there is a 2-approximation algorithm based on minimum spanning tree which we have implemented in this part.

In some cases, we do not find an edge value for direct edges in the MST. in this case, we have used the shortest path distance between these 2 vertices.

.We have further used the below relationship to do our approximations.

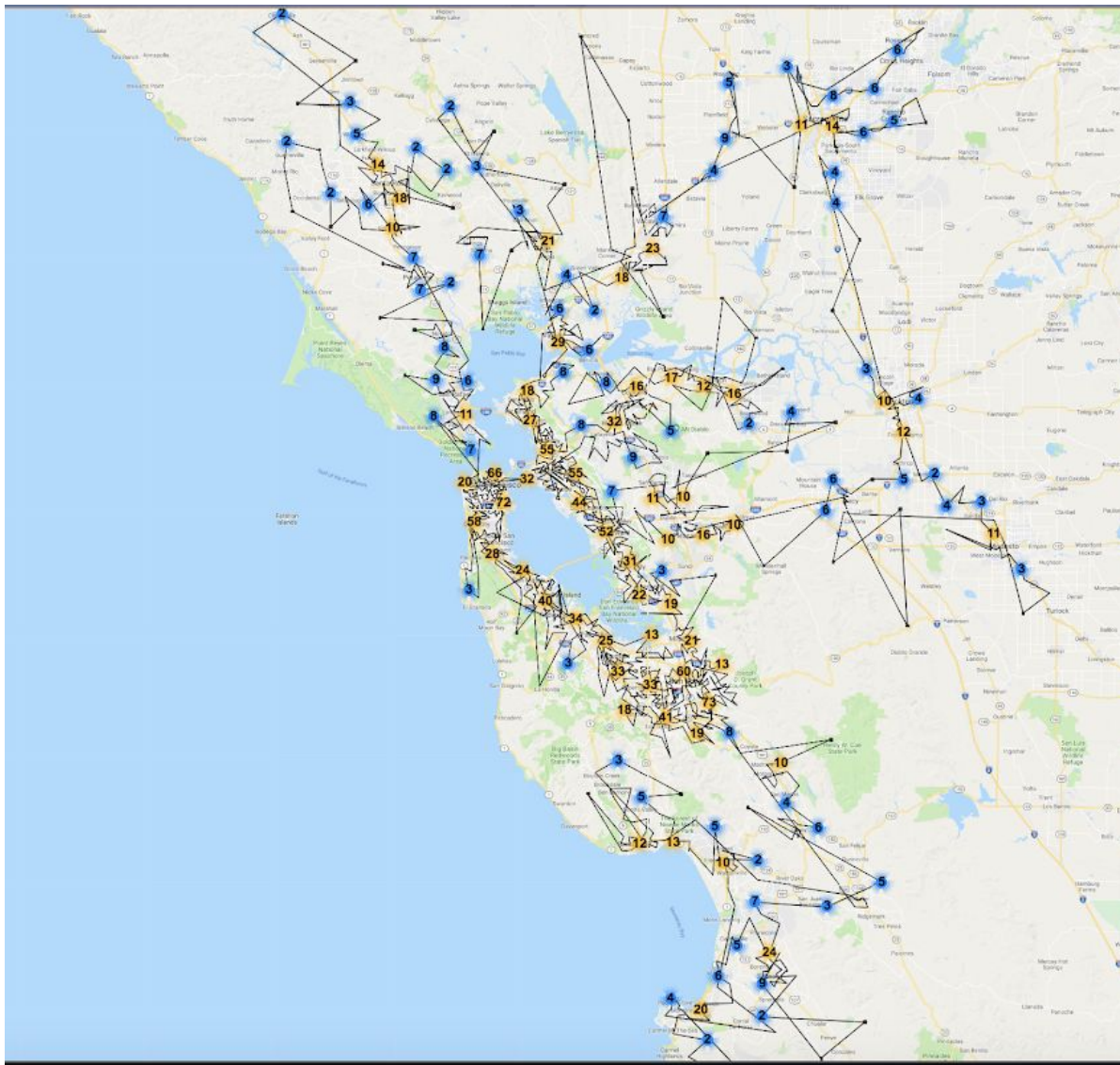
Cost of MST < Cost of Optimal TSP < Cost of 2-approximate TSP < 2 x Cost of Optimal TSP

As min value of Optimal TSP is the cost of MST, we have chosen the Cost of MST and 2 x Cost of MST as the bounds for our 2-approx TSP and we get the below results.

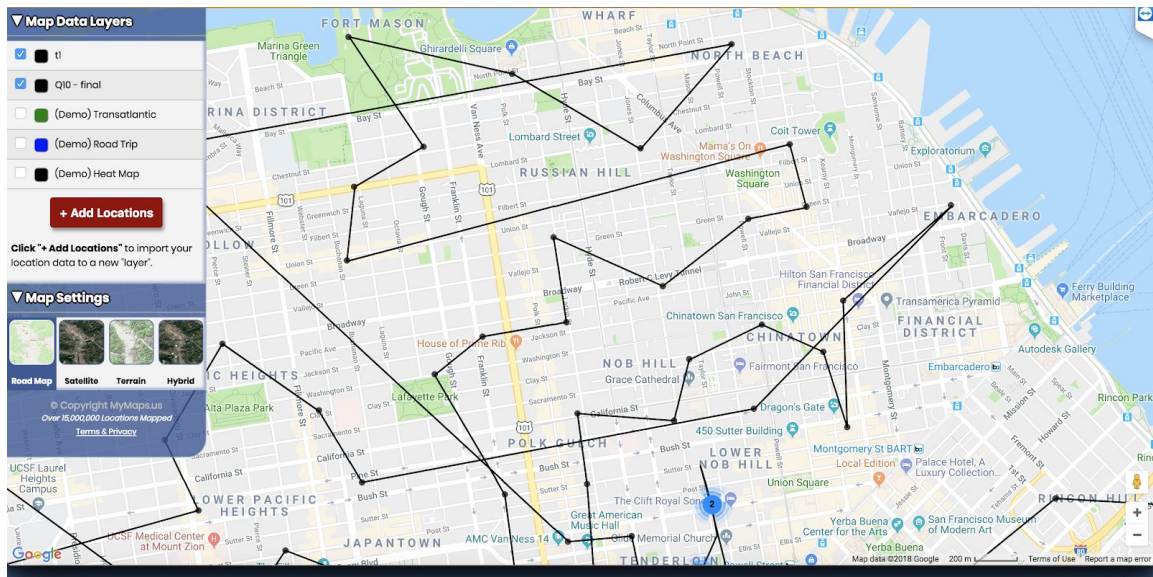
MST Cost	279408.2
Approx TSP Cost	466489.6
2 x MST Cost	558816.4
Ratio Upper Bound (Approx TSP Cost / MST Cost)	1.669563

2.5. Plot the trajectory that Santa has to travel!

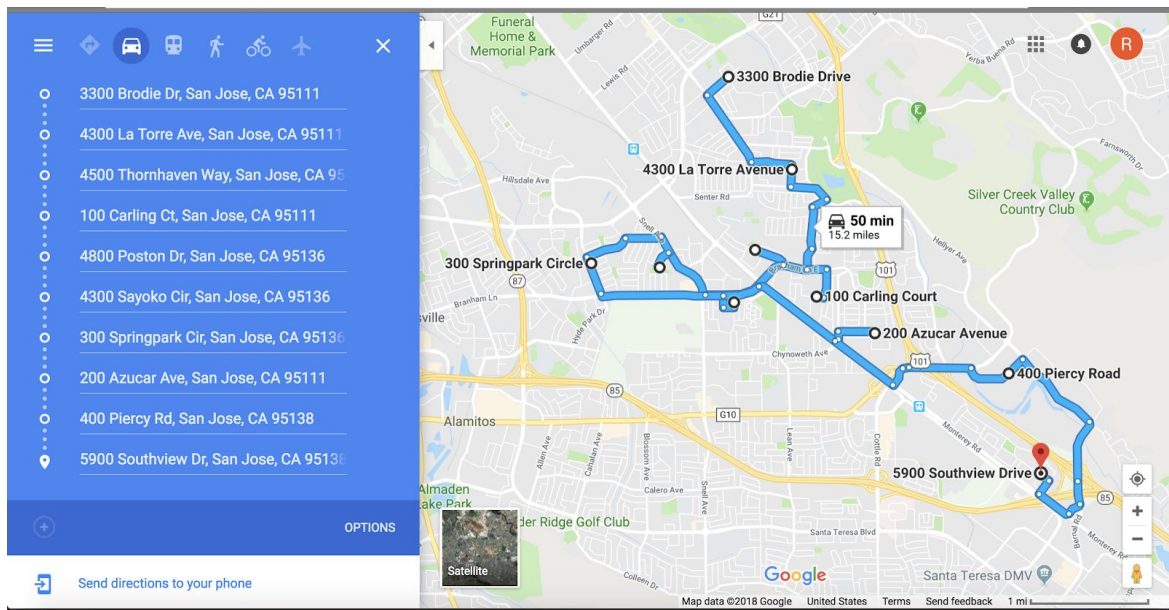
We plot the order in which Santa has to travel to complete the task at hand. We plot the path to be followed and the complete plot of the trajectory is show below.



A more zoomed in version better shows the trajectory followed and stops taken. The figure below shows this. The stops are close by and an order is followed as visible in the figure.



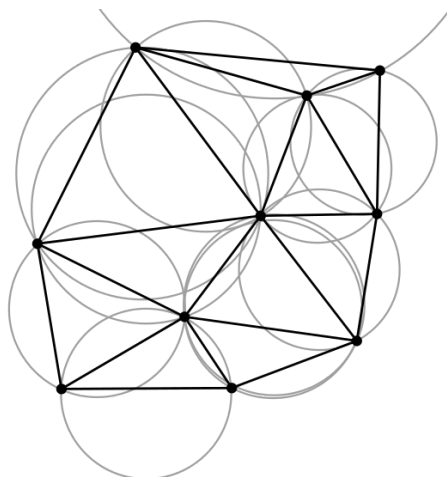
To get an even better understanding, we plot the consecutive points given by our algorithm on Google Maps with multiple consecutive destinations. The figure below shows this result. We see that the stops are properly in order and each closeby stop is covered while minimizing the time as well as covering all points.



3. Analyzing the Traffic Flow

3.1. Plot the road mesh that you obtain and explain the result. Create a subgraph G induced by the edges produced by triangulation

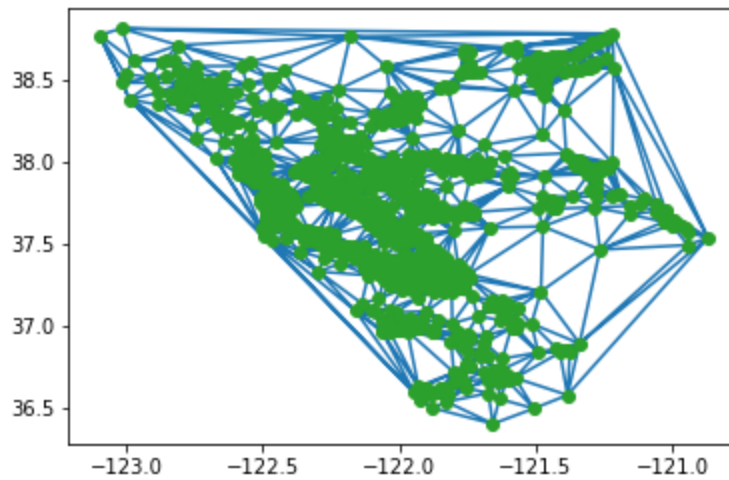
In this part, we apply Delaunay Triangulation to the Uber Network to generate a road mesh. a Delaunay triangulation (also known as a Delone triangulation) for a given set P of discrete points in a plane is a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$. Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid sliver triangles. A sample output of Delaunay triangulation is shown below:



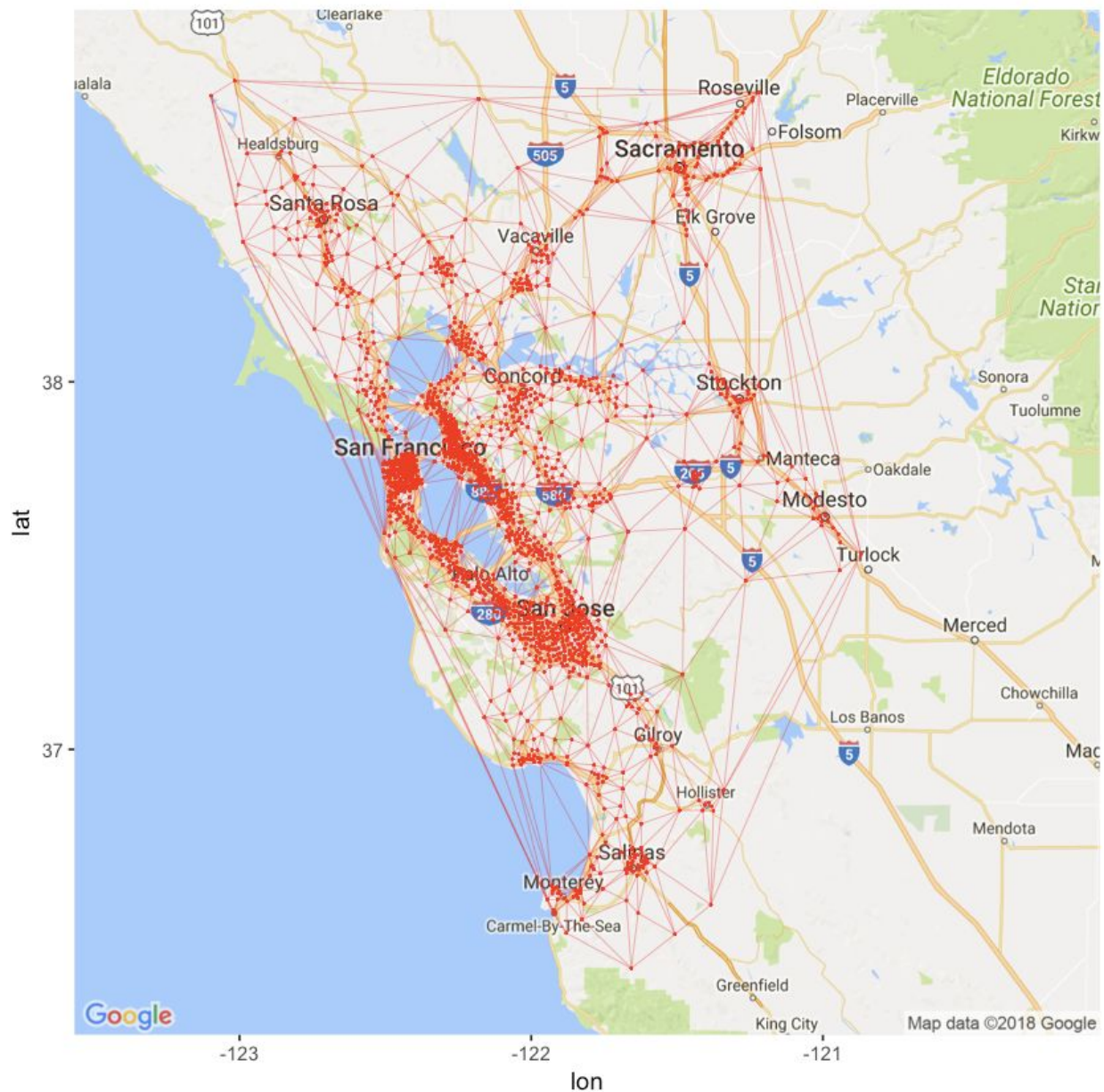
On applying the triangulation to the uber network, we achieve the following road mesh with approximately 5.6k edges:

Number of Vertices	1880
Number of Edges	5627

A plot of the road mesh formed by triangulation is shown below. The road mesh indeed looks like a map of California.



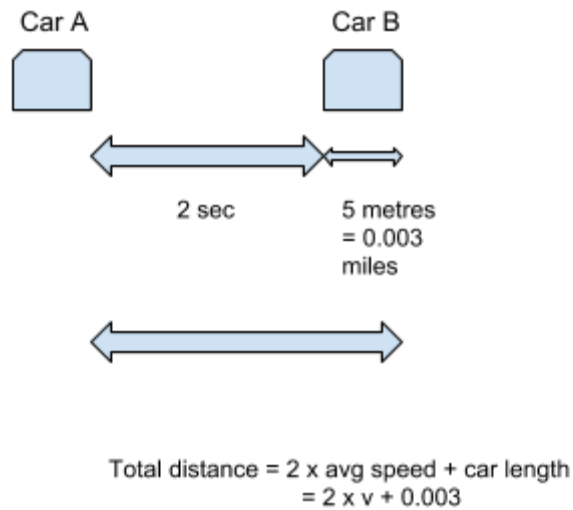
To better visualize the road mesh, we plot the results of the triangulation and our new graph on the map as shown below. One thing that can be easily observed is that the actual roads and the edges formed by triangulation are not the same and might not overlap. We also see some edges over water in the bay area and near Monterey.



3.2. Using simple math, calculate the traffic flow for each road in terms of cars/hour.

To calculate the traffic flow, we need to find out how many cars pass the edge in an hour.

We are given that the length of a car is 5 metres and that it has to maintain a safety distance of 2 sec between cars. This would look like this:



Here v is the average speed which is computed as

$$v = \text{distance between two coordinates} / \text{average travel time}$$

Hence, the time taken for one car to pass will be

$$\frac{\text{total distance}}{v} = \frac{2xv + 0.003}{v} = 2 + \frac{0.003}{v}$$

Therefore, traffic flow will be defined as

$$\text{traffic flow} = \frac{1 \text{ hour}}{2 + \frac{0.003}{v}}$$

This traffic flow is the capacity of the road. As there are 2 lanes on the road, we have multiplied the above value by a factor of 2 to get the maximum capacity of each road.

Statistics of Capacity

Max Capacity	3565.103
Min Capacity	635.7120
Mean Capacity	2754.035
Std. Deviation Capacity	353.256

3.3. Calculate the maximum number of cars that can commute per hour from Stanford to UCSC. Also calculate the number of edge-disjoint paths between the two spots. Does the number of edge-disjoint paths match what you see on your road map?

We used the Max flow algorithm for this task using the maximum capacities of roads we found in the previous parts.

Maximum flow problems involve finding a feasible flow through a single-source, single-sink flow network that is maximum.

The maximum flow problem can be seen as a special case of more complex network flow problems, such as the circulation problem. The maximum value of an s-t flow (i.e., flow from source s to sink t) is equal to the minimum capacity of an s-t cut (i.e., cut severing s from t) in the network, as stated in the max-flow min-cut theorem.

The edge connectivity of a pair of vertices (source and target) is the minimum number of edges needed to remove to eliminate all (directed) paths from source to target. `edge_connectivity` calculates this quantity if both the source and target arguments are given (and not NULL).

The edge connectivity of a graph is the minimum of the edge connectivity of every (ordered) pair of vertices in the graph. `edge_connectivity` calculates this quantity if neither the source nor the target arguments are given (ie. they are both NULL).

A set of edge disjoint paths between two vertices is a set of paths between them containing no common edges. The maximum number of edge disjoint paths between two vertices is the same as their edge connectivity.

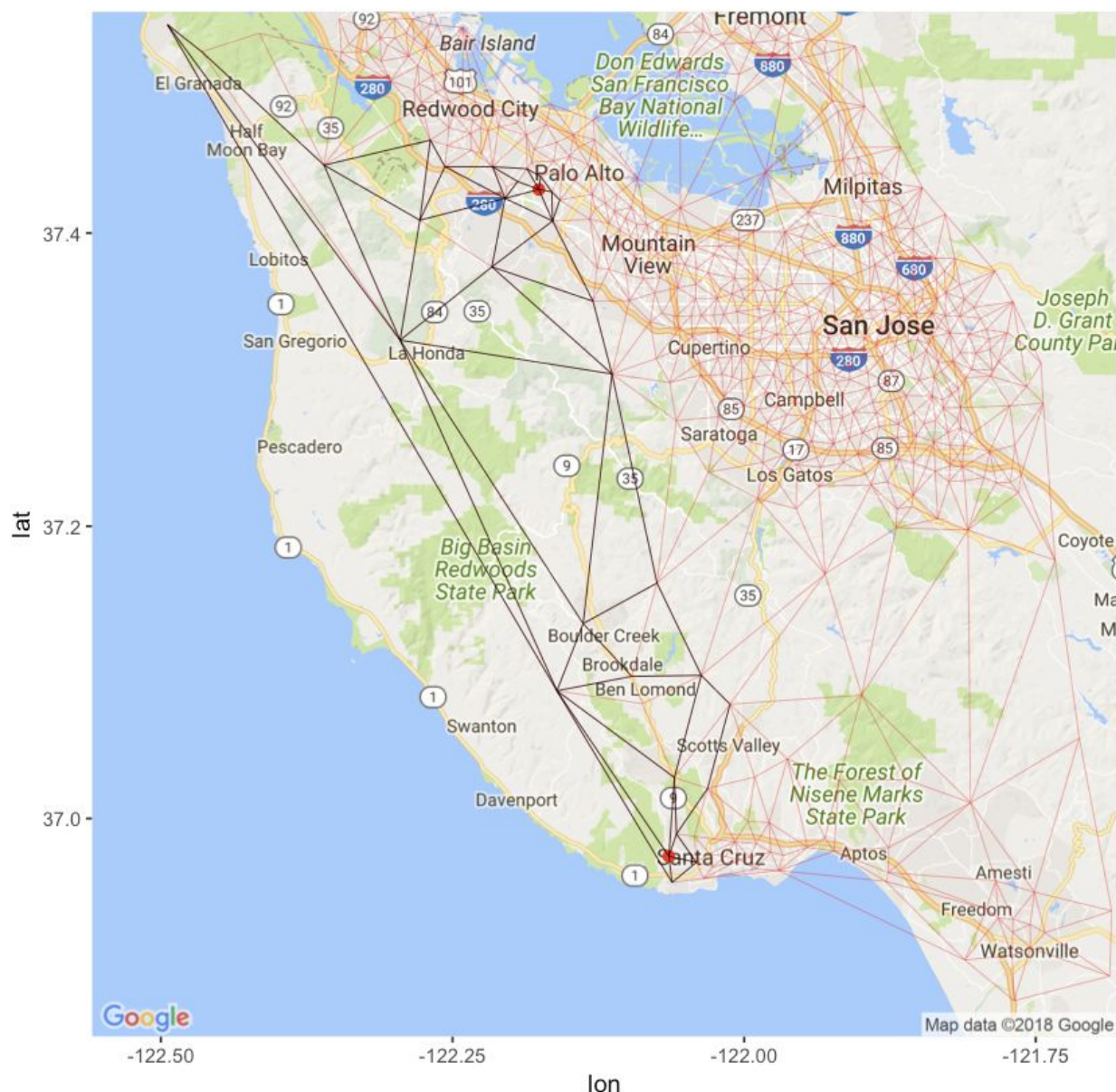
We get the below results for the maximum flow.

Max Flow Value	14747.969
Number of Edge Disjoint Paths	5

The max flow algorithm along with the maximum value also returns the path or flow along each of the edges. We visualize the results of the max flow algorithm on the map. The source and sink is denoted by 2 big red markers

on the map. The red edges are all the edges of the graph. The black edges are the edges involved and contributing to the max flow. With this representation, it is easy to visualize the edges and hence the paths. We can see 5 paths leaving out of the source and 5 paths entering into the sink. Following these 5 paths, we see detect the 5 edge disjoint paths as returned by the algorithm.

Also the original graph generated in question 6 gives total 94 edge disjoint paths. It is a densely connected giant component and hence we get more edge-disjoint paths there as compared to our road mesh generated by triangulation.



3.4. Plot \tilde{G}_Δ on real map coordinates. Are real bridges preserved?

We first compare the original graph max and min times with the graph returned by triangulation. We clearly see that the graph returned by triangulation has a higher max travel time value mostly due to the fake edges and fake bridges. We see a lot of fake bridges on the map in Q11 in the bay and near Monterey. Based on the statistics of the travel time in our new graph, we see that the mean is just around 360 with a deviation of 680. So to remove the fake edges, we filtered based on these statistics and removed the outliers. We only keep the edges which lie around the mean, specifically within 1 standard deviation and remove the edges which have weights (or travel time) above ($\text{meanTime} + \text{stdDeviationTime}$). On filtering based on this method to remove outliers and unreal edges, we deleted 262 edges and so the new defoliated graph contains 1880 vertices(same as before) and 5365 edges.

Comparison of Time with Original

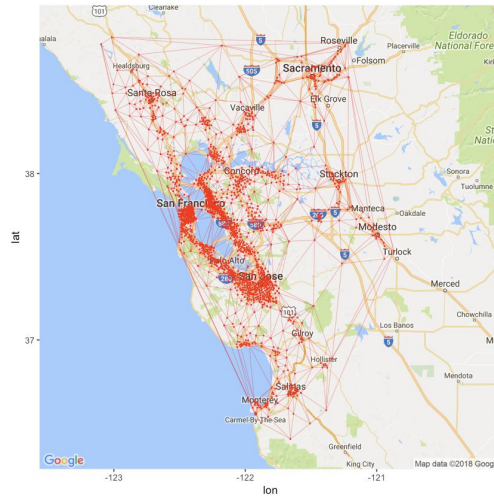
Original Graph Max Time	6439.18
Original Graph Min Time	21.9
G by Triangulation Time	10265.625
G by Triangulation Time	21.9

Statistics of Travel Time in G by Triangulation

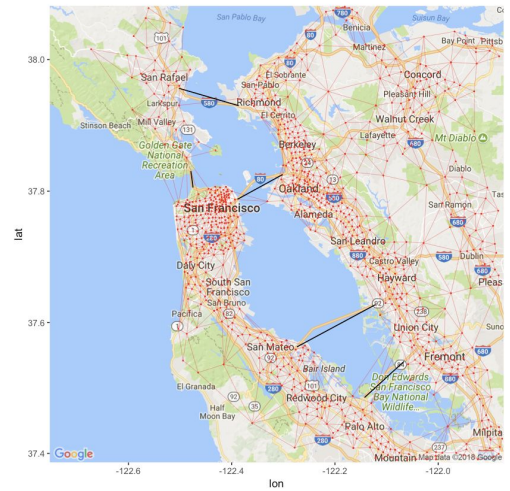
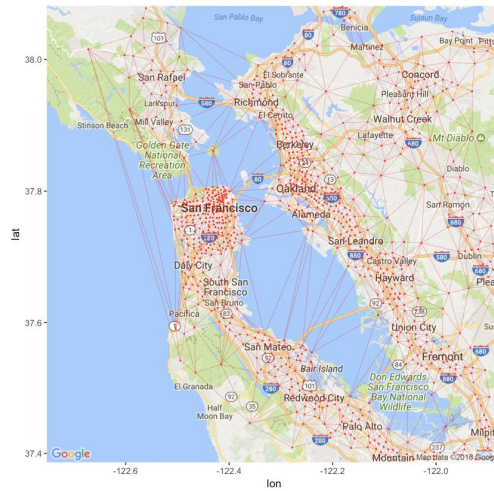
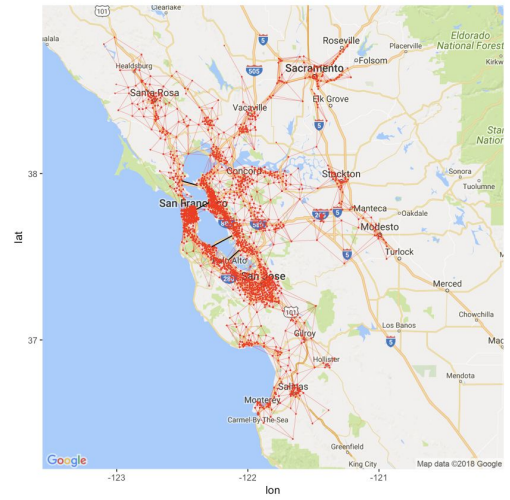
Max Travel Time	10265.625
Min Travel Time	21.9
Mean Travel Time	364.197
Std. Deviation Travel Time	686.260

We visualize these results on the map to get a better understanding. We first show the results in tabular form for easier comparison.

Original Graph Mesh

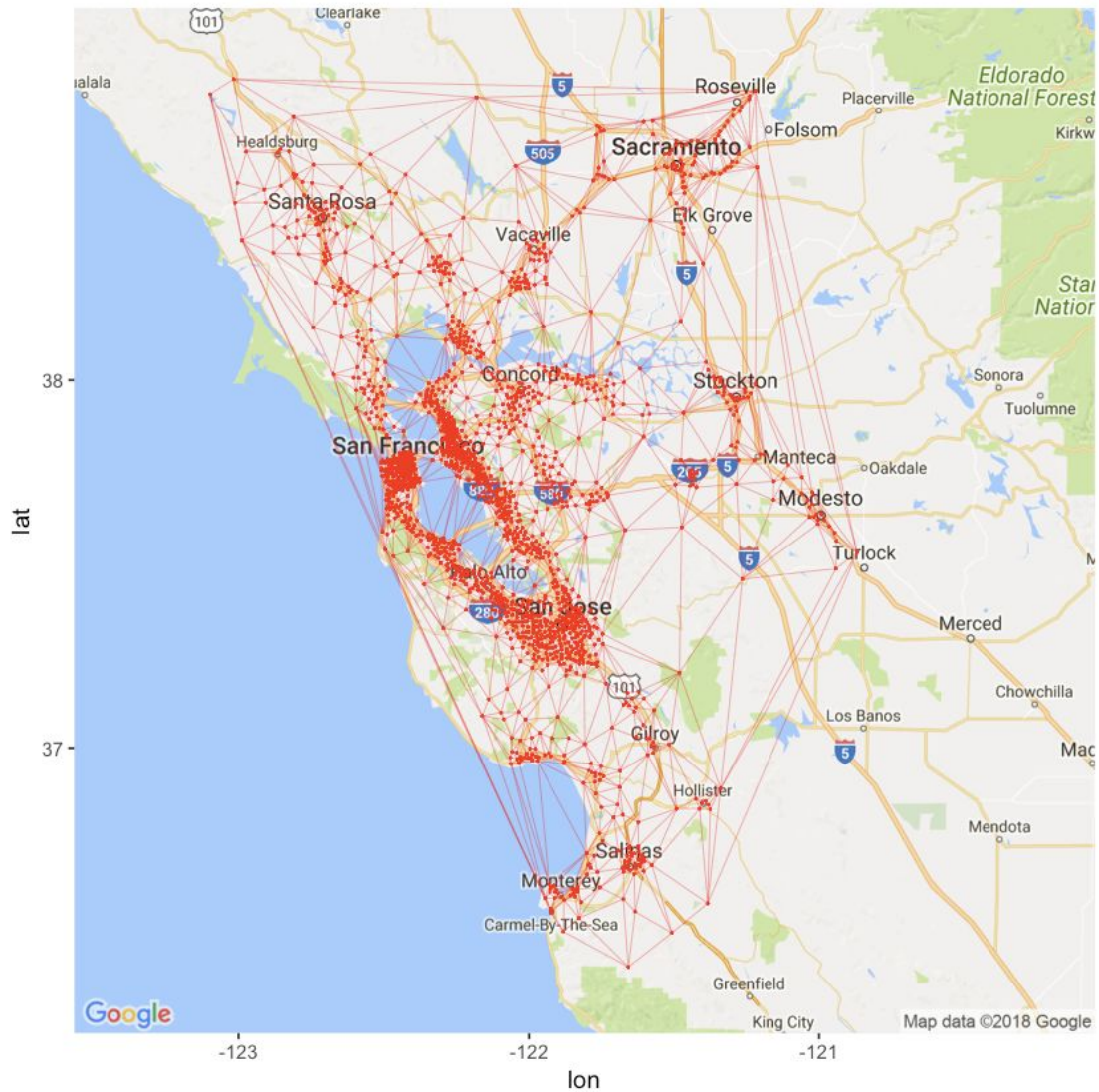


Defoliated Graph Mesh

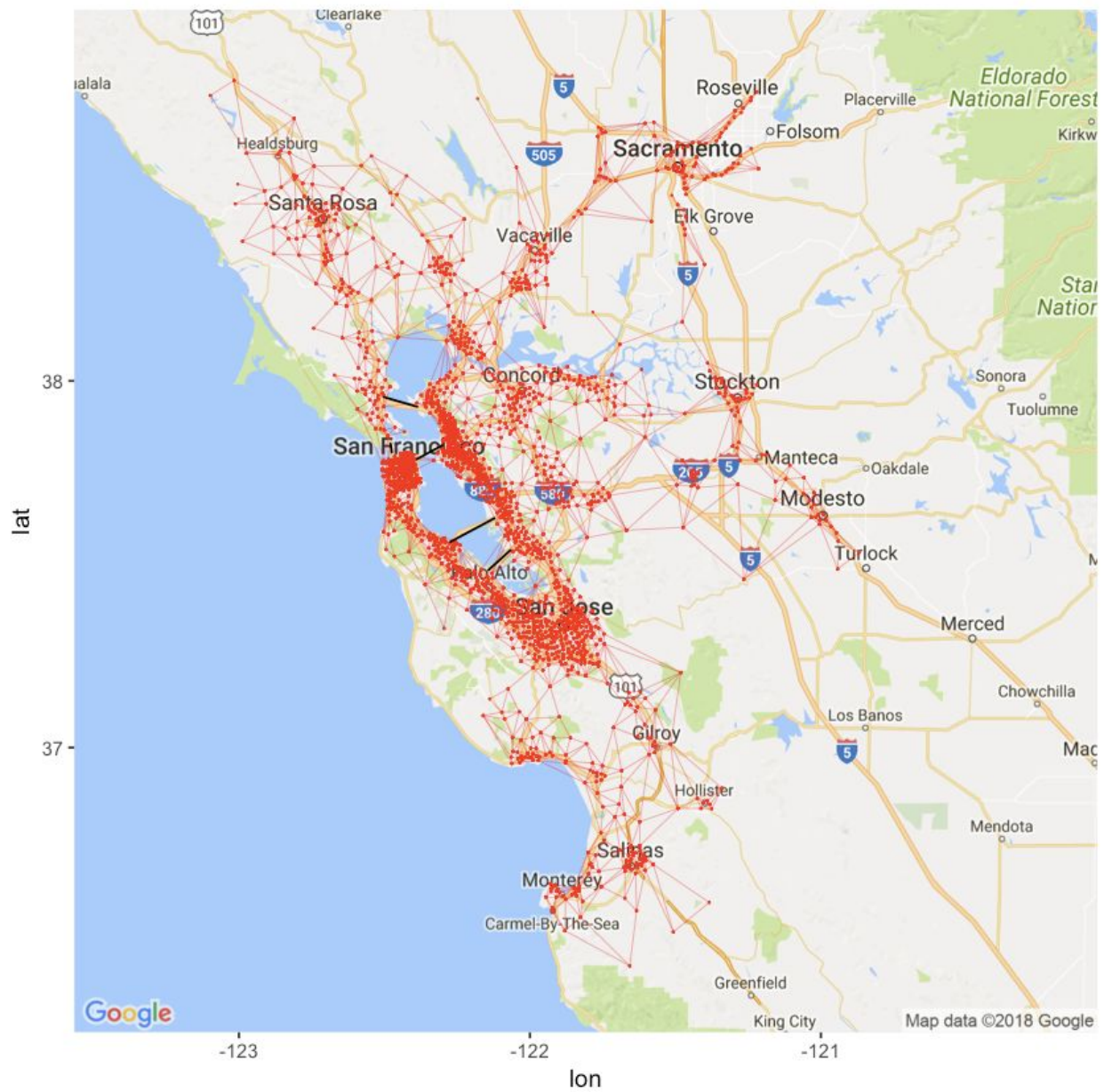


The same results with bigger images for analysis are shown below.

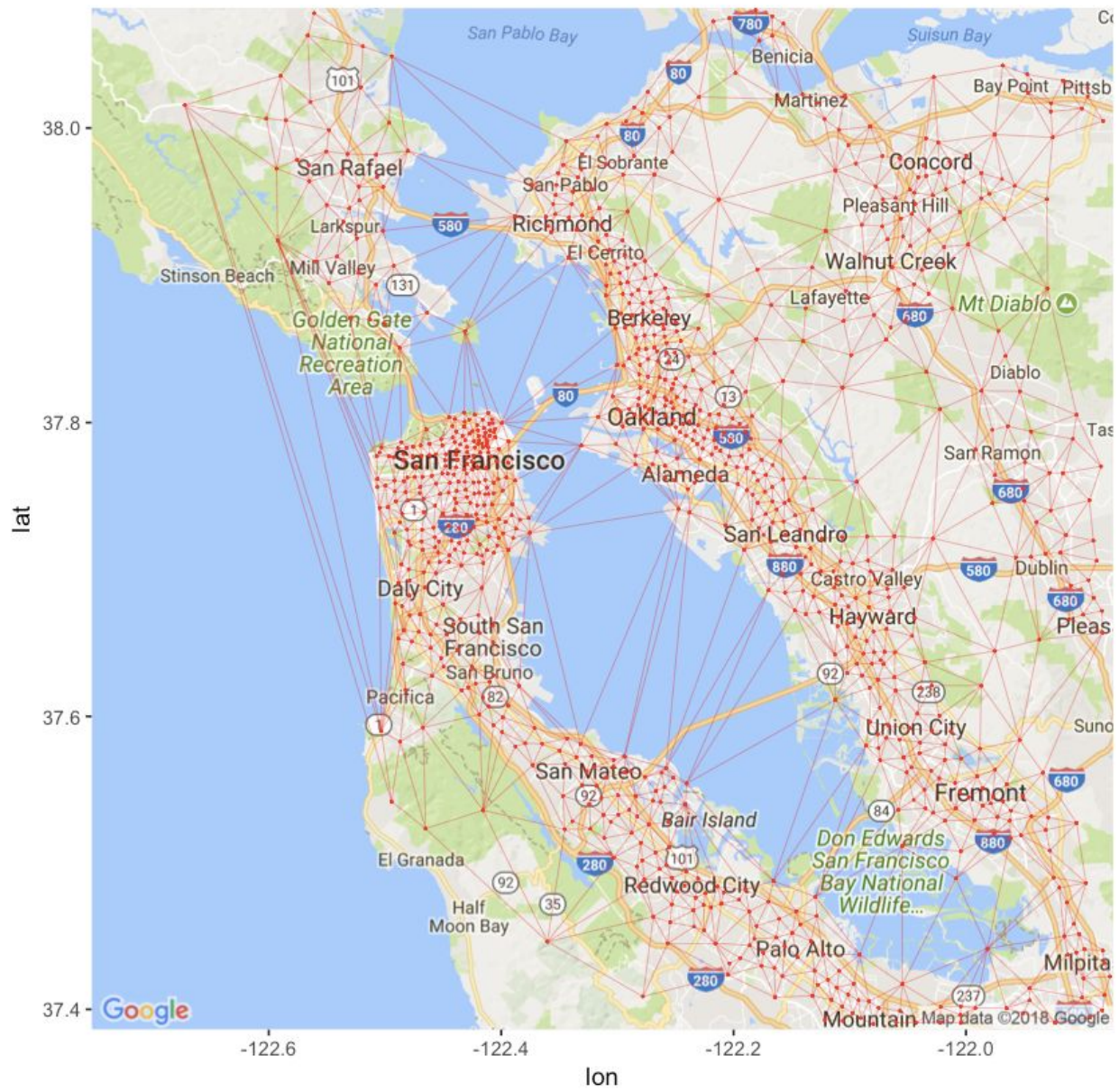
1. Original graph mesh

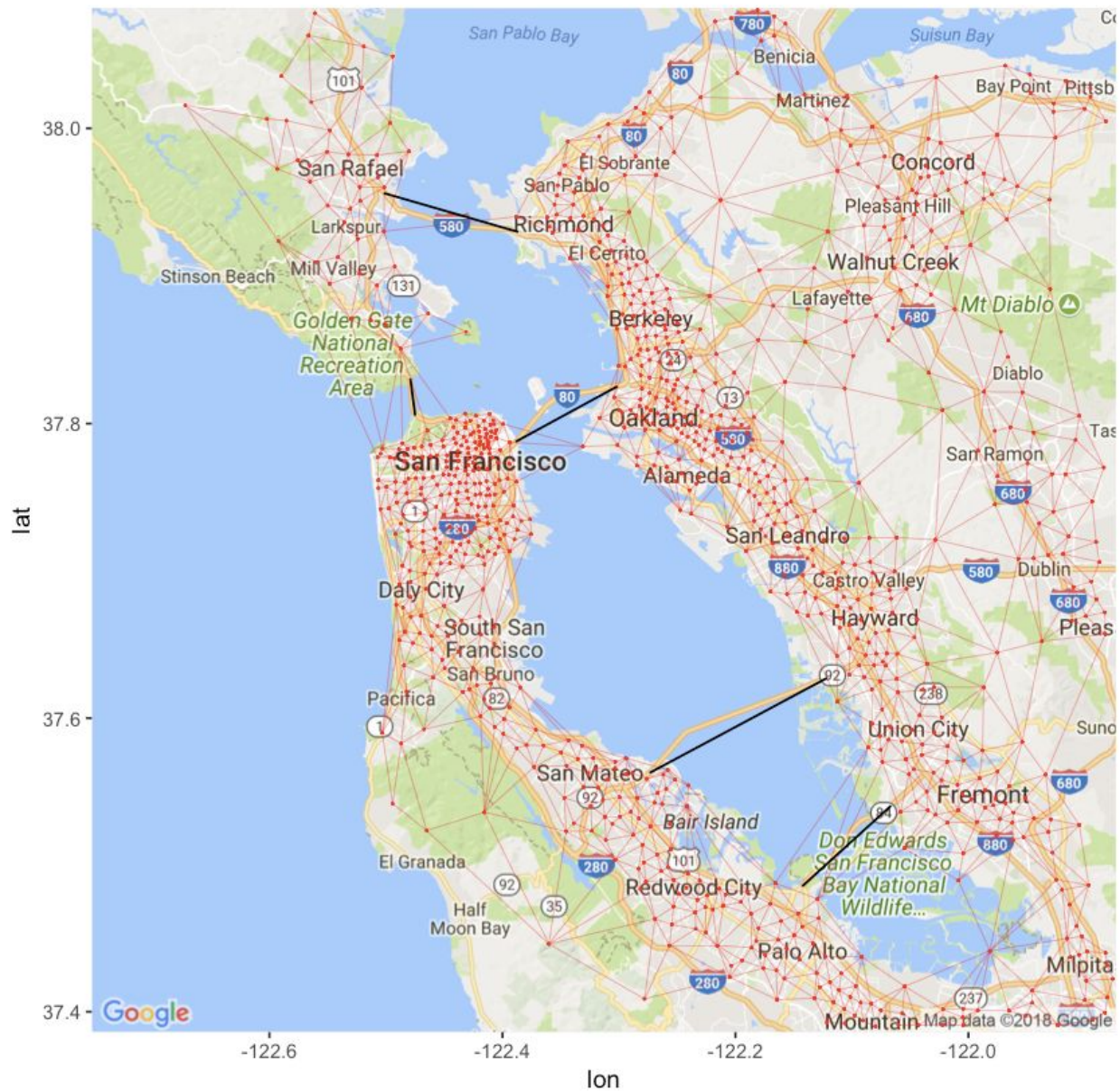


2. Defoliated graph mesh with fake edges/bridges removed



Zoomed in version of the bridges:





The above 2 images are zoomed in on the 5 major bridges in the bay area. The black edges denote these 5 bridges. After removing the fake edges, we can see that almost all (4 out of 5) real bridges are preserved and the other fake edges and routes over the bay have been successfully filtered out.

3.5. Now, repeat question 13 for \tilde{G}_Δ and report the results. Do you see any significant changes?

On repeating max flow algorithm on this new defoliated graph we get the below results.

Max Flow	14747.969
Edge Disjoint Paths	5

We see that we get no changes in the results as compared to question 13. We get the same results for max flow as well as the number of edge disjoint paths. The technique we have used to defoliate the graph involves removal of outliers which do not contribute to the max flow. As the number of edge disjoint paths do not change, the max flow value also does not change.