EE232 Project 5
Graph Algorithms

Team members:
Sonali Garg (104944076)
Shweta Sood(905029230)
Karan Sanwal(205028682)
Ashish Shah(804946005)

Question 1: Provide an upper and lower bound on ρij. Also, provide a justification for using log-normalized return (ri(t)) instead of regular return (qi(t)).

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

As we can see, pij is cross correlation between stock i and j (based on their temporal averaging). Cross-correlation values range between -1 to 1, which is the lower and upper bound respectively.

The paper has used log normalized return (ri(t)) instead of regular return (qi(t))
This has a very beautiful mathematical intuition.
Let's take an example of a stock on an annual basis. It's price is Rs 50 in 2015 and grows to Rs 100 in 2016 and finally drops back to Rs 50 in 2017.

If we look at the average return of stock over the 2 years it is = (100-50)/2 = 25%
This means the stock grew 25% over the 2 years. However, we do know this is False because it's price is same as the original.

Now, let's take the log normal returns = (ln(100/50)+ln(50/100))*100/2 = (69-69)/2 = 0
Hence, **log normal eliminates the positive bias of average returns** by putting all numbers in a log plane. Thus, log return is a better metric than just taking raw returns!

There are other reasons as well:

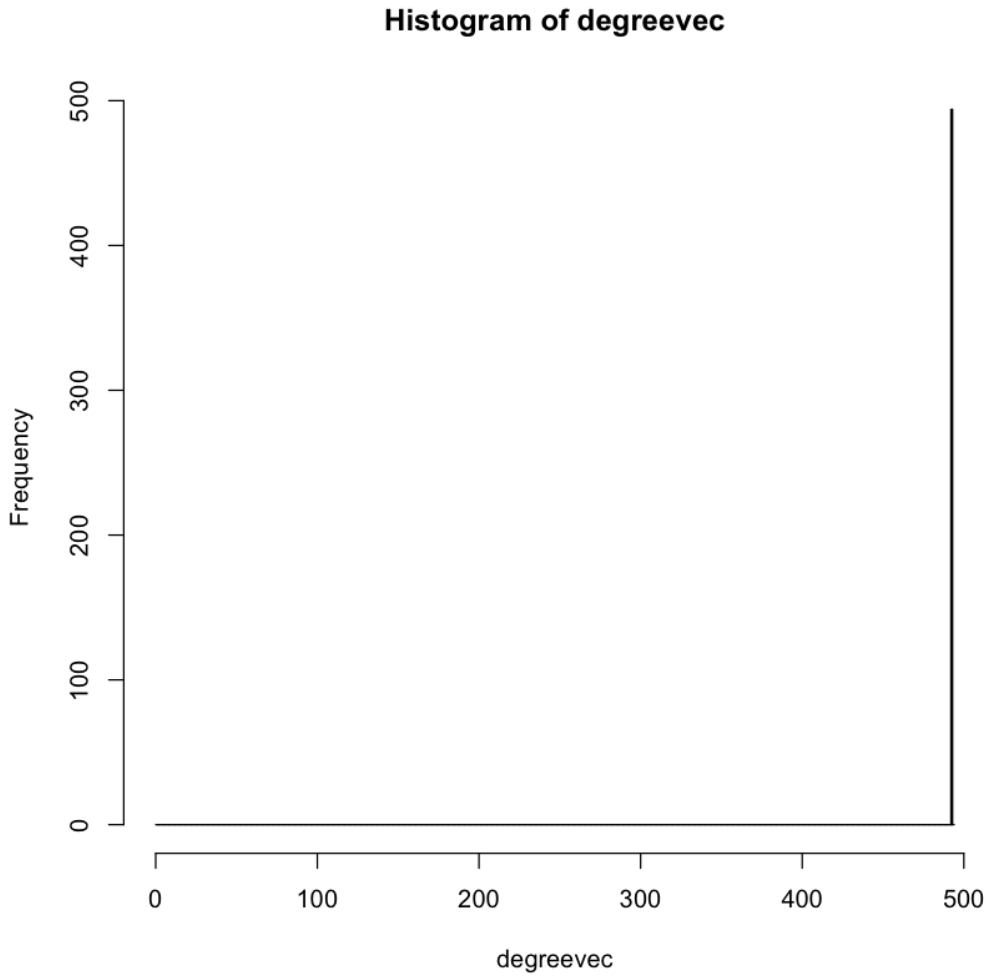
1. log(1+r) ~ r, r<<1 : When returns are smaller than 1, their log normal is close in values to them

2. Multiplication of small numbers may lead to arithmetic underflow. Addition of the small numbers is instead safe. Log gives the flexibility to convert a multiplication to logarithmic addition making it safe!
3. Log normality is a useful transformation as returns will be normally distributed. This in coherence with requirement of major Machine Learning problems.

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

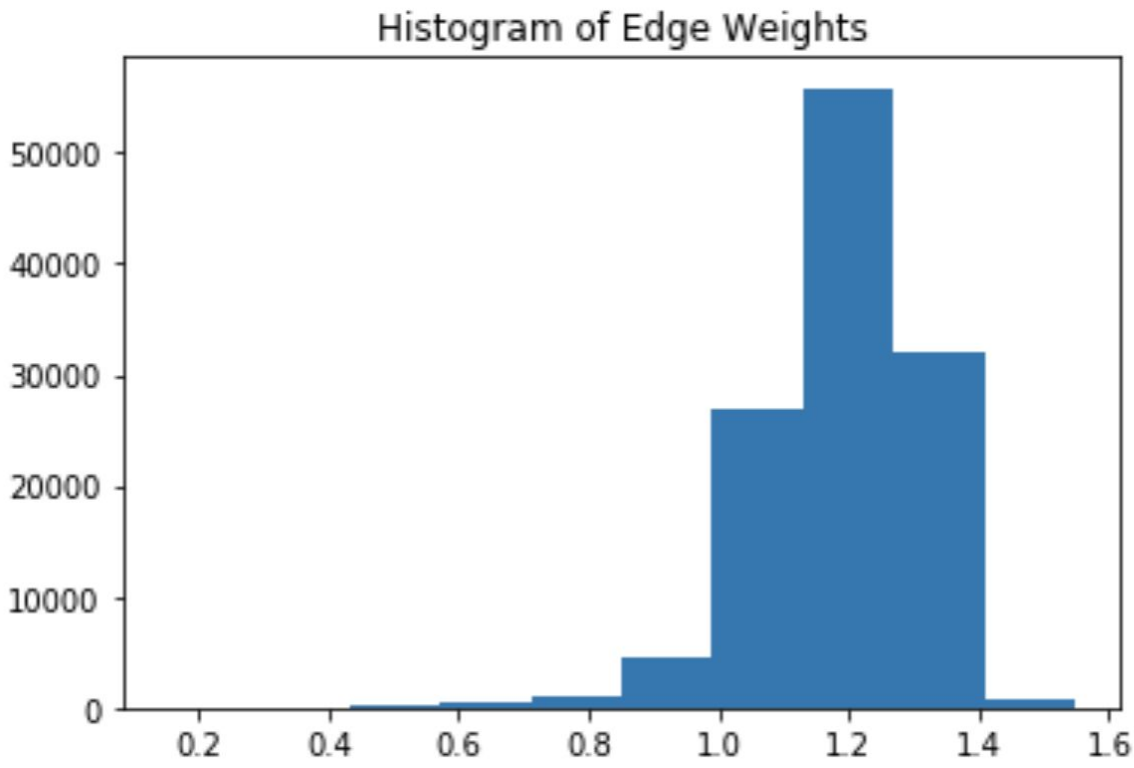
In order to construct a correlation graph, we only considered stocks that had exact same dates. Rest of the stocks were ignored. The edge weights are described by

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



Degree distribution of the correlation graph

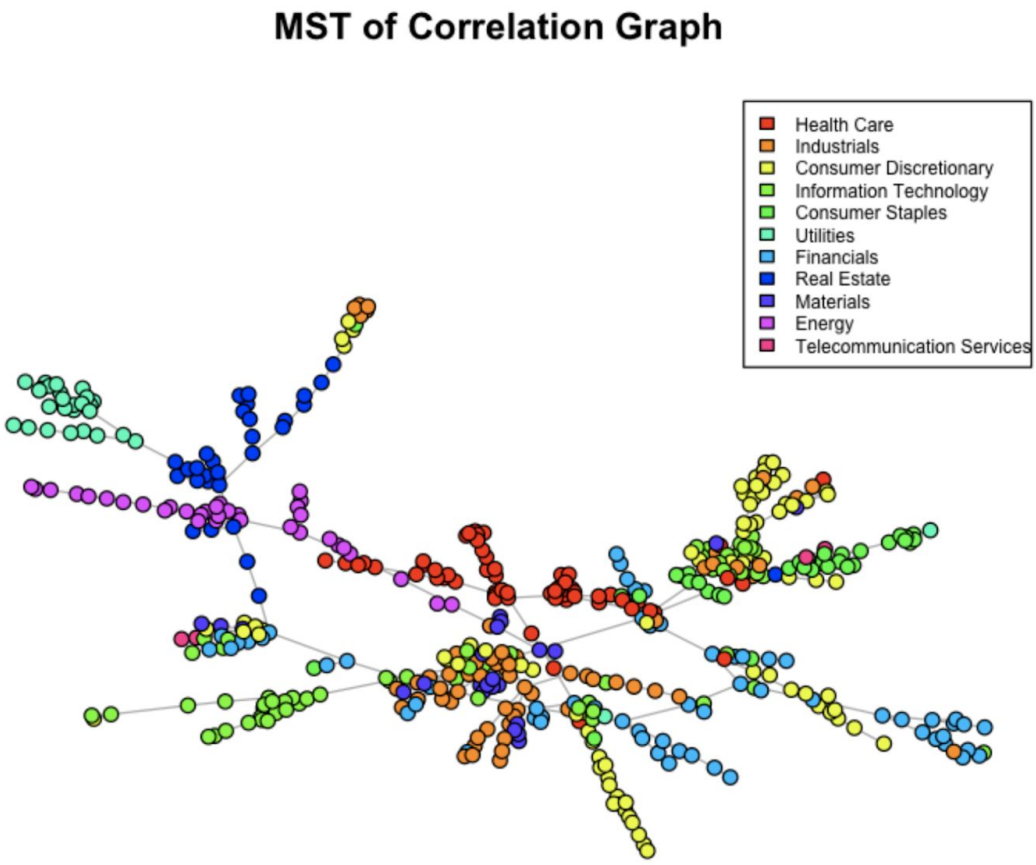
We can see from the degree distribution of the correlation graph that all the stocks have the same degree distribution of 494 as it's a fully connected graph. So every node is connected to 493 other nodes. This is because it's a correlation graph trying to evaluate how closely associated is it to other stocks which can be used for predicting it's behavior based on the similar stocks.



Un-normalized distribution of Edge Weights

Looking at the histogram of edge weights, we notice that the edge weights are in the range of 0 to 2 (also based on the formula described earlier). The most common weights between edges are 1.2-1.4

Question 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.



The above MST of the Correlation Graph is constructed by color coding the nodes based on their sectors.

There are 11 sectors in all: Health Care, Industrials, Consumer Discretionary, Information Technology, Consumer Staples, Utilities, Financials, Real Estate, Materials, Energy, Telecommunication Services

We notice that the nodes of the same color are clustered together in the MST, representing a vine like structure. The stocks represented by each clusters' nodes belong to the same economic sector. The MST structure can be used to identify different economic groups in the stock market. This happens because investors tend to have same investing strategy for stocks in the same economic sector.

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

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

where S_i is the sector of node i. Define

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

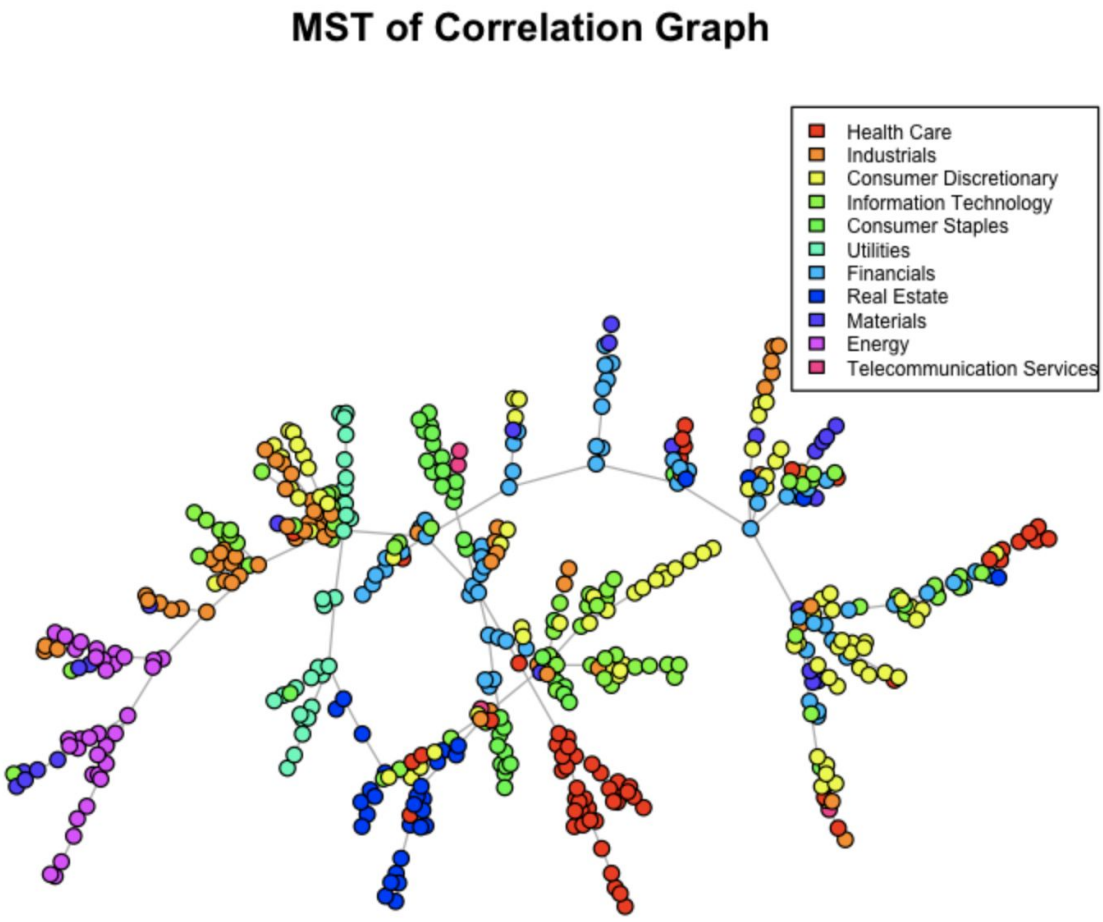
where Q_i is the set of neighbors of node i that belong to the same sector as node i and N_i is the set of neighbors of node i. Compare α with the case where

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

For Case 1: Value of alpha is 0.828
For Case 2: Value of alpha is 0.116
We notice that the performance is much better in Case 1 than in Case 2. This is expected because of the way we have defined probability of a node belonging to a sector. In Case 2, we are calculating this probability, based on all vertices of the graph,ie, of all the vertices in the graph how many belong to a particular sector.

In contrast, Case 1, has a more refined manner of finding the probability. It looks at the nodes in the neighborhood of the node whose sector has to be found. Out of the nodes in neighborhood, how many belong to the same sector as the node under consideration? The neighborhood of the node is more likely to represent nodes in the same sector as we saw from the vine clusters of the MST. Thus, it is a precise representation of the nodes similar to that node. Hence, it will provide a more accurate calculation of the probability.

Question 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.



Now, we extracted the MST from the correlation graph based on weekly data. MST of the vine cluster obtained here looks similar to vine cluster obtained using daily data. However, here the vine clusters aren't that refined/ well segregated. To compare this MST, with the previous one, we again did the analysis of performance of the probability of a node belonging to a sector. Here, for Case 2, alpha was same as the previous analysis: 0.116. For Case 1 (which is a more refined manner of defining probability), this alpha was 0.719. We realize that the performance falls for this MST. This can be reasoned based on how we constructed the MST. Since, we only use weekly data to build the correlation graph, it is a more summarized representation of a neighborhood, albeit not a very accurate one. Thus, we observe a loss in accuracy.

Question 6: Report the number of nodes and edges in G.

In this part we created the cleaned graph G as described in the question. First graph was made with edges from sourceid to destid(undirected) with mean travelling time as weight (where month = 12) . Then for each vertex we added attributes location and display name using the geo boundaries(json) file. Then we just found and retained the largest connected component(gcc). The gcc had 538984 edges and 1880 nodes. Finally, duplicate nodes were removed.

Solution:

Number of nodes=1880
Number of edges=311802

Observation:

This entire process makes a lot of sense. First, using mean travelling time as the edge weight makes sense as generally nowadays that is seen as more important metric than simply distance(cab services like Uber require the driver to take the route that takes the shortest time by default, not shortest distance). This is because traffic conditions etc. can make short paths take a long time and people prefer to save time. Secondly, taking the greatest connected component is obviously vital as otherwise some nodes would be unreachable from some other nodes and the problems that follow(like travelling salesman) would not be possible(at least not without modifications).

Another thing to notice is that the gcc is not fully connected. The number of edges are a lot less than they should be in a fully connected case given 1880 nodes.(1880 chose 2 i.e. 1766260). This information becomes useful in Q9 where knowing that some nodes may not have directly edges between them, we place checks and take shortest distances whenever edge does not exist.

Question 7: 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?
Solution:
In this part we build the MST of graph G.
According to Wikipedia, MST is "a subset of the edges of a connected, edge-weighted (un)directed graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight."
Looking through some of the edges we observed that for most of them, the two endpoints seem to be fairly close by. Some of the edges we saw were:

'3300 Brodie Drive, South San Jose, San Jose' -> '4300 La Torre Avenue, South San Jose, San Jose'

'3300 Brodie Drive, South San Jose, San Jose' -> '3700 McLaughlin Avenue, South San Jose, San Jose'

'3300 Brodie Drive, South San Jose, San Jose' -> '400 Ginkgo Court, South San Jose, San Jose'

'1400 Calle Alegre, South San Jose, San Jose' -> '1200 Redmond Avenue, South San Jose, San Jose'

'1400 Calle Alegre, South San Jose, San Jose' -> '5600 Park Crest Drive, South San Jose, San Jose'

Observation:
The results do seem intuitive. Searching on google maps, we find that all these are 3-5 minute away by car, i.e. fairly close by. This makes sense as MST tries to connect all vertices together, without any cycles and with the minimum possible total edge weight (and here, edge weight is the mean travel time, so we are analysing in terms of time) and hence, intuitively a lot of locations that are close by should be connected, which is the case in our results too.

Question 8: 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.
Solution:
In this part, we consider triangle inequality based on the mean travelling time(it makes sense as that is what we use in place of actual distance in this problem). We find all the triangles in the graph and then randomly sample for 1000 of them.
On randomly sampling 1000 triangles, we find that **91.2%** of them follow triangle inequality,i.e. If x,y,z are the weights(mean-travelling times in this case) then x+y>z , x+z>y , y+z>z , all 3 conditions are satisfied in 912 out of 1000 random samples.

Observation:

In general, triangle inequality states that for any triangle, the sum of the lengths of any two sides must be greater than the length of the remaining side, i.e. distances always follow triangle inequality(unless all 3 points lie on the same line).
However, here we consider triangle inequality in terms of mean travelling time. We observe from the result that while triangle inequality is satisfied in most cases, in ~10% of the cases it is violated. This makes sense as it is possible that going from A to B and B to C is faster than going directly from A to C. This could possibly be due to reasons like traffic and so on. It is common for people to take indirect longer routes as compared to direct ones because they are faster.

The 2- approx TSP in the next part only guarantees upper bound when triangle inequality is satisfied. Since a pretty high percentage of triangles satisfies triangle inequality, we should be able to get a good performance.

Question 9: Find the empirical performance of the approximate algorithm:
ρ =
Approximate TSP Cost
Optimal TSP Cost

Solution:

Approximate TSP Cost:
The travelling salesman problem is a famous NP Hard problem. The problem is : “Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.”
Since no polynomial time algorithm is known to solve it, approximations are used to solve it. These approximations rely on the problem instance satisfying triangle inequality. We saw in Q8 that a pretty high percentage of triangles satisfies triangle inequality, hence we should be able to get a good performance. (Even if some don’t it will only lead to increase in upper bound, which way it would still be an upper bound)
In the approximate algorithm, first we randomly chose a node as the starting and ending point of the walk. This node also serves as the root for the MST. Then we construct an MST from the graph. Finally, we use dfs to find the preorder traversal of the mst and add the root node at the end of it to make a complete tour.

Optimal TSP Cost:
Since, finding optimal cost is NP hard, we take the sum of the weights of the edges of MST as an approximation of the optimal cost. This works because we know that :

MST cost < optimal TSP cost < approximate TSP cost < 2 times MST cost
Since, we are looking for an upper bound, decreasing the denominator is fine, it will still give an upper bound(though perhaps not as strict).

On applying the 2-approximate algorithm for TSP, the results seems intuitive. Looking at the first 10 nodes in the traversal we get :

"3300 Brodie Drive, South San Jose, San Jose" **->** "4300 La Torre Avenue, South San Jose, San Jose" **->** "4500 Thornhaven Way, Edenvale, San Jose"**->**"100 Carling Court, South San Jose, San Jose" **->** "200 Azucar Avenue, Edenvale, San Jose" **->**“400 Piercy Road, Edenvale, San Jose"**->** "5900 Southview Drive, Edenvale, San Jose"**->**"Mimosa Way, South San Jose, San Jose" **->** "7100 Vía Romera, South San Jose, San Jose"
-> "100 Cheltenham Way, South San Jose, San Jose"

All these streets seem to be 3-8 minutes(according to google maps) away from its previous street. Hence, in the approx TSP path nearby nodes(in terms of mean travelling time) are next often next to each other i.e. the travelling time between node i and i+1 in the TSP path is mostly low.

Finally, we compute the ratio as asked:
Approx TSP cost= 464758.3
Optimal TSP cost=279408.2
Therefore, **ρ =1.6634**

Observation: In general, in 2-approx TSP we know that the cost of the approximate algorithm is less than the cost of the full walk(which is at most twice the cost of mst) and any optimal solution to TSP can never be less than cost of mst(by definition of mst). This is because in preorder walk 2 (or more) edges of full walk are replaced with a single edge, the length of which by triangle inequality has to be less than the sum of the lengths of the 2 edges. Hence, we know that the algorithm is 2 approximate.
In this instance, since we showed that most mean travelling times satisfy triangle inequality, we expect a good performance and hence, similar upper bound. We get the ratio as 1.6634 which is less than 2 and in the expected range of results. Also, we use the MST cost as the optimal cost in the ratio equation, which is acceptable as we are looking for an upper bound, decreasing the denominator is fine, it will still give an upper bound(though perhaps not as strict).

Question 10: Plot the trajectory that Santa has to travel!

Solution:

We plot the trajectory that Santa has to travel. We use the location attribute to get the coordinates corresponding to the nodes in the tour we get in Q9(corresponding to the 2-approx solution for TSP). In this we plot the tour we get from our algorithm directly,(not the one that would actually be followed in the graph as some edges are missing in the graph and we have to compute shortest path for those.(as we were told that in office hours)

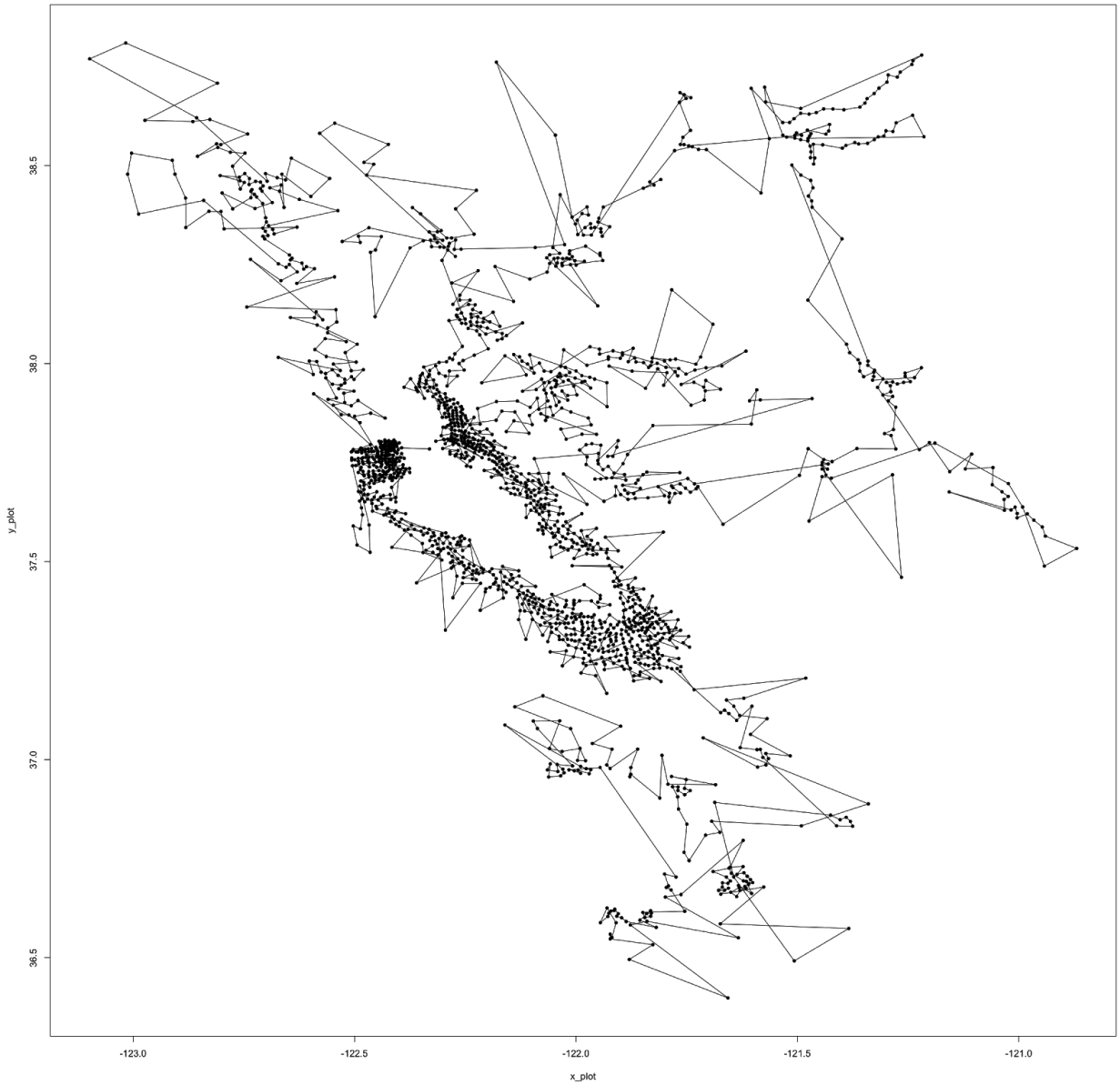


Fig: Santa's Trajectory

Observation: We notice that there seems to be complete tour and that no edges are being repeated(there are 1880 edges in the graph as expected), which is what was expected as in 2-approximate TSP we basically do a Eulerian walk. Many of the edges seem to be very short, which also makes sense given problem statement. Some of the messiness of the graph could possibly be attributed to the fact that TSP was done based on travelling times and we are plotting actual distances on this graph by using coordinates.

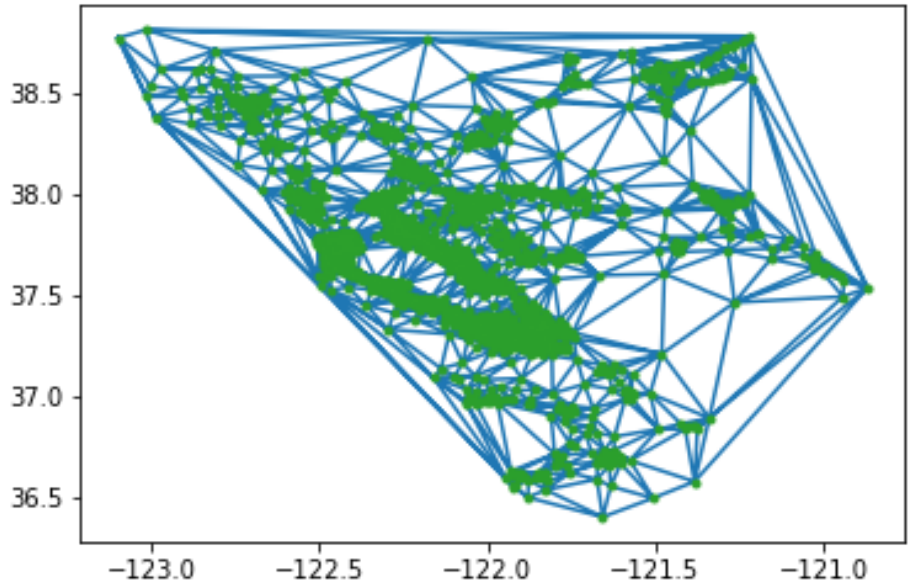
PS: In case the trajectory of the actual path taken in the graph(including the nodes that are present in the shortest path if there is no direct edge between 2 nodes) is required, it is present in the ipython notebook called "Q6_7_8_9_10_final" as output in 44th cell.

Question 11: Plot the road mesh that you obtain and explain the result. Create a subgraph G induced by the edges produced by triangulation.

For this part, we had to estimate the map of roads without using the actual road dataset given in the question. We had to use the coordinates of the Uber dataset for this question.

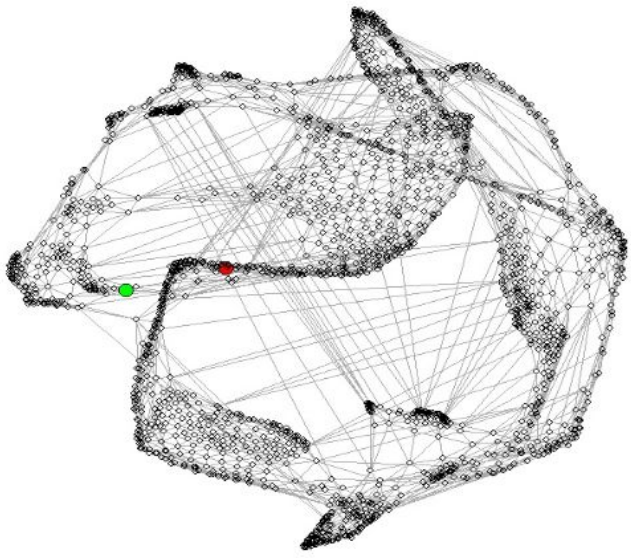
We were asked to use the Delaunay triangulation for this question in order to estimate the road map. The Delaunay algorithm finds a set of triangles for a set of points such that no discrete point in the set is inside the circumcircle of the triangle.

It maximizes the minimum angle of all the angles of the triangles in the triangulation.



Number of edges in Delaunay triangulation = 5627
Number of vertices in Delaunay triangulation = 1880

We also created a subgraph induced by the edges produced by the triangulation.



The green and red markers show the destination and source respectively.

Question 12: Using simple math, calculate the traffic flow for each road in terms of cars/hour.

- **Each degree of latitude and longitude - 69 miles**
- **Car length 5 m = 0.003 mile**
- **Cars maintain a safety distance of 2 seconds to the next car**
- **Each road has 2 lanes in each direction**

Assuming no traffic jam, consider the calculated traffic flow as the max capacity of each road.

We used the following formula for calculating the traffic flow. Traffic flow is nothing but the number of cars per hour. We can consider any arbitrary point in the road and see how many cars pass it in an hour.

From the above assumptions in the question, a car length is 0.003 mile. And we need to leave a gap of 2 seconds between cars.

Time=Distance/Speed
Time taken by 1 car = 2/3600 + 0.003/Speed

We know Speed=Distance/Time

We get the distance through the end points of the road using the Euclidean distance formula and time through the mean travel time between end points as we calculated above.

$$d(\mathbf{p},\mathbf{q}) = d(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Distance is multiplied by 69. And speed will be in miles/hour.

After using the above equations, the formula we get for traffic flow(number of cars per hour) is 1/(2/3600 + 0.003/Speed) OR Speed/(2*Speed/3600 + 0.003)

Through this we get traffic flow for 1 lane. Through the questions assumptions, we multiply the result we get from this with the number of lanes.

Final formula we used is **Speed/(2*Speed/3600 + 0.003) * number of lanes**

Question 13: 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 calculate the number of edge disjoint paths between the source (100 Campus Drive, Stanford) and destination (700 Meder Street, Santa Cruz). An edge disjoint path is a path that does not share a edge with any other path.

We had a total of **5 edge disjoint paths**.

We calculate the maximum flow on the edge of the disjoint paths of the graph through sum of cars per hour on the edge disjoint paths of the graph.

Maximum number of cars that can commute = 29783.13

Yes, we can see that the number of edge disjoint paths matches with what we see in the road map.

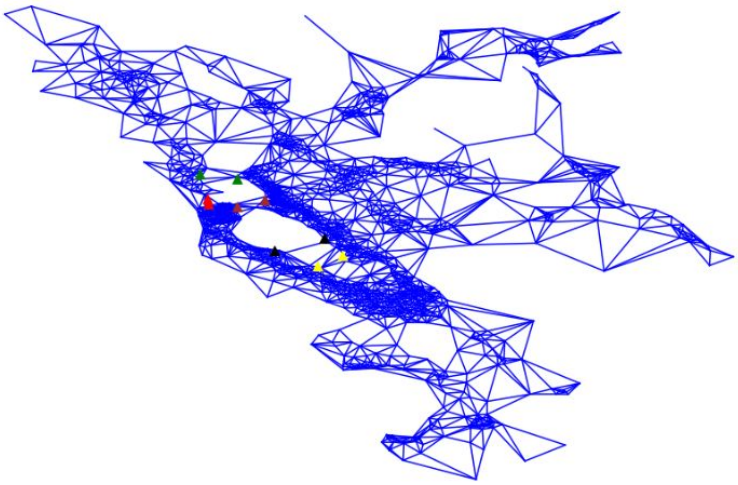
Question 14: Plot ~G on real map coordinates. Are real bridges preserved?

In this question, we defoliate the graph by applying a threshold on the mean travel time of roads and remove a few roads, to result in the below graph. This eliminates the fake bridges with abnormal mean travel times in the original graph.

I marked the below graph with small triangles to show the real bridges.

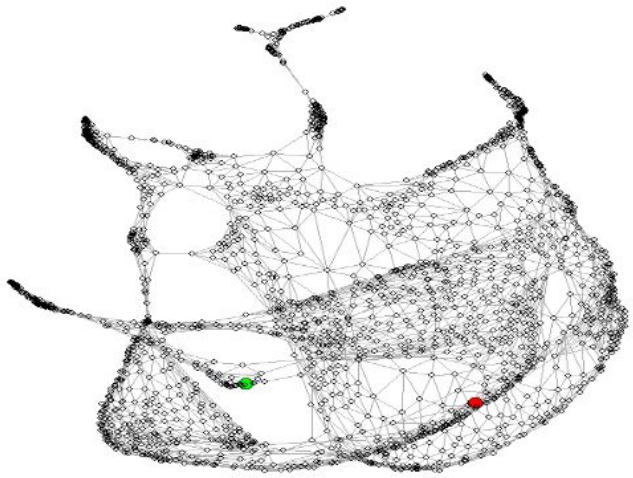
Dumbarton Bridge	Yellow
San Mateo Bridge	Black
Golden Gate Bridge	Red

San Rafael Bridge	Green
San Francisco - Oakland Bay Bridge	Brown



We can see that, yes, the above graph preserved the real bridges.

Question 15: Now, repeat question 13 for ~G and report the results. Do you see any significant changes?



From repeating Q13 we get the following results.

Number of edge disjoint paths = 5
Maximum number of cars that can commute = 29783.13

Also, to answer the latter question, No, we don't see any big significant changes. We notice that the observations and results are the same as that of Q13 above.

We can reason whis by seeing that the defoliated graph removed the fake bridges with the really outlier travel times but it didn't remove the edges that are part of the disjoint paths.

Hence, the maximum flow is calculated with the disjoint paths, so the answer for commute is also same.