Large Scale Social Network : Models and Algorithms Project #4

Due on June 12th, 2020 at 5:00pm

Professor Roychowdhury, Vwani

Wang, Yin, He

Question 1

If we regard $r_i(x), r_j(x)$ as two random variables X, Y, then the equation of ρ_{ij} can be reformulated as

$$\rho_{ij} = \frac{E[XY] - E[X]E[Y]}{\sqrt{E[X^2] - (E[X])^2}\sqrt{E[Y^2] - (E[Y])^2} \tag{1.1}$$

which indeed is the formulation of Person correlation. Hence, the upper and lowers bound on ρ_{ij} are 1 and -1 respectively. The usage of log-normalized return $(r_i(t))$ rather than regular return may result from the asymmetrical value domain of the function $q_i(t)$. Notice that $p_i(t)$ is always larger than zero, meaning that the minimum value for $q_i(t)$ is -1 while the maximum value for it is $+\infty$. That is to say, the temporal average $\langle q_i(t) \rangle$ is much more sensitive to the positive value, which might result in the inappropriate correlation coefficient. Let us construct two return arrays for illustration. Let $A_1 = [-\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, -\frac{1}{2}, 4]$, $A_2 = [1, 1, 1, 1, 2]$. If we use the regular return, we find out the correlation for this two arrays is positive. However, according to the our intuition, the correlation for these series should be negative. By employing log-normalized return, we will get what we expect.

Question 2

The correlation coefficient, ρ_{ij} for two temporal series can be easily obtained from the equation given in Question 1, then the weight of edge between node i and node j can be derived. However, during the calculation of these weights, there exists a slight problem resulting from the mismatch length of the closing price arrays. Thankfully, even though the start time of the record of these stocks is not the same, the end time of the record is exactly at 05-12-2017. To compute the correlation coefficient of two arrays, we have to make the length of them identical. Two possible way to make this modification: One is to pad the shorter array with zero; The other is to eliminate the elements before the start time of shorter array. We would like to choose the latter method to compute the correlation coefficient and then get the weight for two stocks. The result is shown as Figure 1.

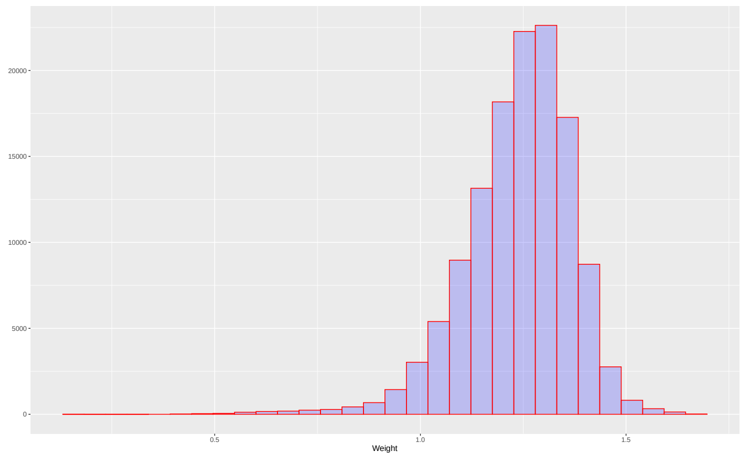

Figure 1: The histogram for the un-normalized distribution of edge weights

Question 3

After obtaining the correlation coefficient for every pair of two stocks, we can arrange these coefficients as an adjacency matrix so as to construct a weighted undirected graph. Then minimum spanning tree is applied

Question 3 continued on next page...

to the graph and the result in shown in Figure 2.

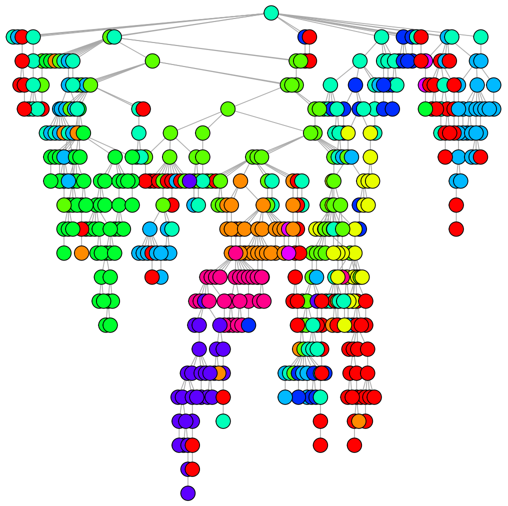

Figure 2: The minimum spanning tree derived from the correlation graph.

By color-coding the nodes of the tree and plot the graph with tree-like layout, we can apparently see the vine clustering pattern. For any node of the tree, the neighborhoods of it mainly belong to the same sector as itself. The appearance of this pattern can be explained from the characters of the algorithm. Since the minimum spanning tree is trying to minimize the sum of edge weights of the spanning tree,the edges with large weights are preferred to be eliminated during iteration. In other words, the edges of vertex i in the MST have relatively smaller weights compared to all edges with node i in original graph. Notice that the weight of the edge between node i and node j is negatively correlated to the coefficient ρ_{ij}. That is to say, the neighborhoods of the nodes in the tree mostly have large correlation with the center nodes. According to our intuition, the stocks in the same sector are always influenced by the similar latent factors, leading to the high correlations between them. So the phenomenon mentioned above can be easily observed from the plot.

Question 4

First, let us compute the value of α for the first case. What we should to is to loop over all the nodes in the generated MST, calculate $P(v_i \in S_i)$, sum it up and average over the nodes. The result of α is 0.7839. The second case of α actually does not depend on the algorithm result but the distribution of the original data. The result for this α is 0.0031. If there is one randomly chosen stock has lost its sector and we try to recover the label of it by randomly choosing a neighbor of it and assign the sector the same as the neighbor, then the equation for α can be seen as the accuracy of our model. Notice that α for the first case can be interpreted as the accuracy of the clustering result based on MST while α for the second case is the accuracy of the original fully connected graph. The difference between these two α demonstrates the great performance of our clustering result based on MST.

Question 5

Inspecting the csv files given, we can find that the data of the stock price have been eliminated the weekend and national holidays automatically. That is to say, we just need to judge whether the related date is Monday or not. By the help of *weekday* and *identical* inherent functions of R, we can successfully derive all the Monday's data from three years' temporal arrays. Applying similar method mentioned in Question 2, 3, we can obtain the clustering result based on *MST* for the Mondays' data, which is shown in Figure 3.

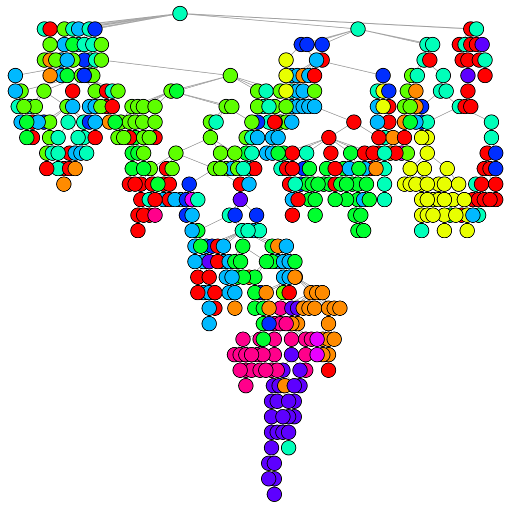

Figure 3: The minimum spanning tree derived from the correlation graph for Mondays' data.

At first glance, we can easily find that the color distribution of the nodes in Figure 3 is more scattered than what we derived in Question 3. The grape-like structures are apparently shown in Figure 2 but it is not the case in Figure 3. To formally evaluate the performance of clustering on *MST* for Mondays' data, we repeat the steps taken in Question 4 and the resulting α is 0.6248, which is consistent with the phenomenon we observe from the plot. The degrading α can be understood from the omission of data. Since the available data we use to predict the relationship between pairs of stocks is reduced, the correlation coefficients are less convincing than the original ones, making the prediction based on the sector label of neighborhood less accurate, which indeed is measured by α.

Question 6

Starting from this question, we are going to use dataset of Travel Times by Month (All Days) of 2019 Quarter 4 for Los Angeles area provided by 'Uber Movement' website. By downloading the dataset from the website, we get a csv file named 'los-angeles-censustracts-2019-4-All-MonthlyAggregate.csv'. In addition to this dataset, in order to better understand the correspondence between map IDs and areas, we also provided the Geo Boundaries file from the same website, which named 'los-angeles-censustracts.json'. Now, by the instruction from the project handout, we take the following steps for this question:

1. Extract only December data from the dataset.

Question 6 continued on next page. . .

2. Merge duplicate edges by averaging their weights
3. Keep the largest connected component of the graph
4. Add the mean of the coordinates of each region's polygon corners as an attribute to the corresponding vertex.

The resulting graph has its nodes corresponding to the locations and undirected weighted edges correspond to the mean traveling times between each pair of locations. For this question, we need to report the number of nodes and edges in the graph G. According to our result, the number of nodes is 2649, and the number of edges is 1004955.

Question 7

In this question, we are going to build a minimum spanning tree (MST) of the graph G. To build the MST is easy by using Python's igraph library. We just use the spinning_tree() function provided by the library. And we get the number of nodes is 2649 and the number of edges is 2648. By just looking at this result, we can see that $E = V - 1$, which corresponds to the property of MST. Next, we take a look at the street addresses of the two endpoints of some edges. Unfortunately, this dataset does not provide detailed street addresses, instead it provides only Census Tract IDs in Los Angeles area. Thus, it is very difficult to location the precise street addresses without considering the precise Latitude and Longitude information. However, our solution to this question is to lookup the Census Tract IDs online and see which ID corresponds to which area in Los Angeles. By using this website `https://data.lacounty.gov/widgets/rv2f-zsc7`, we can find out the corresponding area by matching the IDs. And this website `https://geohub.lacity.org/datasets/enriched-la-county-census-tracts-2015/` maps the Census Tract onto the actual map of Los Angeles, so that we can actually see the location of each Census Tract. Table 1 below is the result of some edges we inspected:

Endpoint 1(ID)	Endpoint 2(ID)	Area of point 1	Area of point 2
554001	554002	Bellflower City	Bellflower City
461700	460800	Pasadena City	Pasadena City
302201	302202	Glendale City	Glendale City
407101	407002	La Puente City	La Puente City
433401	433402	El Monte City	El Monte City
543603	294410	Carson City	Los Angeles City
482001	530700	Monterey Park City	Monterey Park City
460800	463800	Pasadena City	Pasadena City
269100	217001	Los Angeles City	Los Angeles City

Table 1: Endpoints ID and their corresponding areas in Los Angeles.

First, by the theory of MST, we know that the tree can connect all nodes in the graph with the lowest total weight. By looking at this table, and by using the corresponding mapping mentioned above, although we do not have precise addresses, we can still see that every pair of endpoints are close to each other, thus the results are intuitive and consistent with the property of MST.

Question 8

In this question, we are going to first random sample 1000 triangles and determine what percentage of the

Question 8 continued on next page. . .

1000 random sampled triangles satisfy the triangle inequality. Recall that the triangle inequality in graph theory refers to the definition that $E_W(u,v) < E_W(u,w) + E_W(w,v)\ \forall u,v,w \in V$ where $E_W(u,v)$ represents the weight of the edge between node u and node v. This inequality means that the graph has no short cuts. In our case, the vertices will be the points on the map, and we will check that the sum of the lengths of any two sides of the triangle is greater than the length of the third side, and the length in this case is the weight of the edge. Our result shows that 98.2% of the triangles satisfy the triangle inequality.

Question 9

Next, we want to find an approximation solution for the Traveling Salesman Problem (*TSP*) on this graph G. We will apply the 1-approximate algorithm to find the solution. Recall that the 1-approximate algorithm is defined below as:

1. Find the minimum spanning tree T under $[d_{ij}]$
2. Create a multigraph G by using *two copies* of each edge of T.
3. Find an Eulerian walk of G and an embedded tour.

To summarize this algorithm, the first step is to, in our case, build the *MST* from the original dataset. And then we need to create a graph that has two copied of each edge, since we have the undirected graph, both the edges should be the same. The final step is to find a Eulerian walk of graph G. Recall that a Eulerian Walk (Path) is a path in the graph that visits every edge exactly once, and Eulerian Circuit is a Eulerian Path which starts and ends on the same vertex. In this question, we are going to compute the upper bound on the empirical performance of the approximate algorithm, which is defined as

$$\rho = \frac{\text{Approximate TSP cost}}{\text{Optimal TSP cost}} \tag{9.1}$$

In this case, the approximate *TSP* cost is the total weight computed from the Eulerian graph, and the optimal *TSP* cost is the sum of the weight from the *MST*. From our result, $\rho = 1.73$.

Question 10

In this question, we will plot the trajectory that Santa has to travel. In particular, we use the mean coordinate information to plot the trajectory. In the plot, x-axis is the Longitude and y-axis is the Latitude. The resulting plot is shown below as Figure 4.

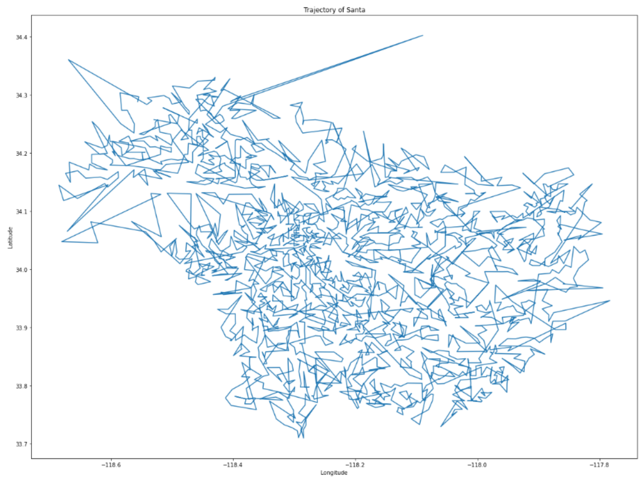

Figure 4: The optimal trajectory that the Santa travels for the gift delivery.

Question 10 continued on next page...

Question 11

We used the *scipy.spatial.Delaunay* to perform Delaunay triangulation on the vertices of G (the cleared graph) and drew the vertices and edges on the flat longitude/latitude plane as Figure 5. It seems like the shape of Los Angeles, but not quite accurate.

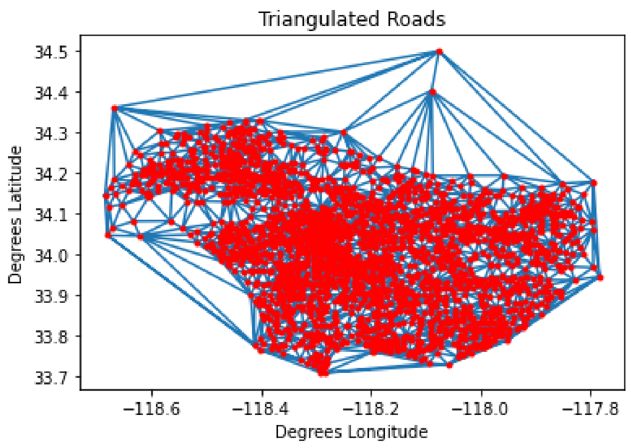

Figure 5: The road mesh obtained by Delaunay triangulation

Question 12

For a car, we set its velocity as $v = \frac{d}{t}$ $mile/h$, then the number of cars remain on a d mile road should be

$$N = \frac{d}{0.03 + v/1800} = \frac{1800d}{54 + v}$$

It takes t hours for these cars to pass the road, so the traffic flow should be

$$q = \frac{1800d/t}{54 + v} = \frac{1800v}{54 + v}$$

Notice that each road has 2 lanes in each direction (4 lanes totally), so the aggregate maximum flow should be:

$$q_{max} = \frac{7200v}{54 + v} \tag{12.1}$$

If consider one direction way only (like what we did in Question 13 and Question 15), then the result should be $\frac{3600v}{54+v}$.

Question 13

To compute the maximum flow and the number of edge-disjoint paths from Malibu to Long Beach, we then use the *max_flow* and *edge_disjoint_paths* functions in *iGraph* of R and the computation results are shown in Table 2.
To verify that the number of paths that do not intersect the edges matches what we saw on the road map, we looked at the enlarged road maps of Long Beach and Malibu. Using black dots to indicate their positions, we can plot the triangulated roads near Malibu and Long Beach as Figure 6 and 7.

Question 13 continued on next page. . .

Lane	Traffic flow	Aggregate traffic flow(maximum flow)	Number of edge disjoint paths
L_1	1892.043	9683.597	4
L_2	2632.522		
L_3	2439.271		
L_4	2719.761		

Table 2: Max flow and disjoint paths from Malibu to Long Beach

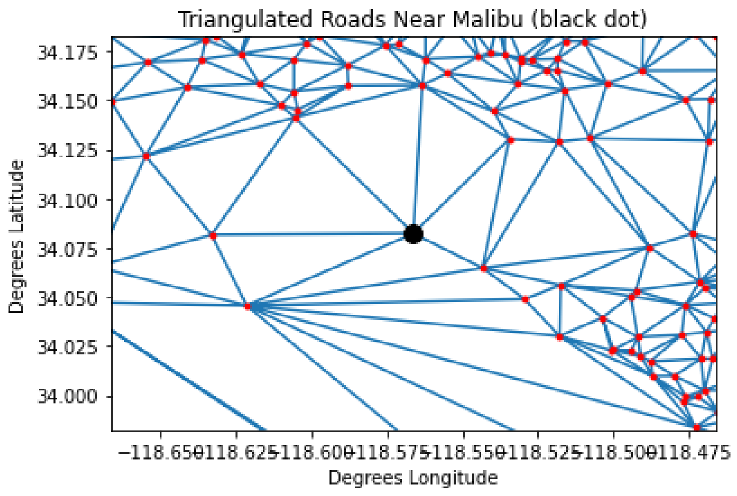

Figure 6: Triangulated roads near Malibu (represented by black dot)

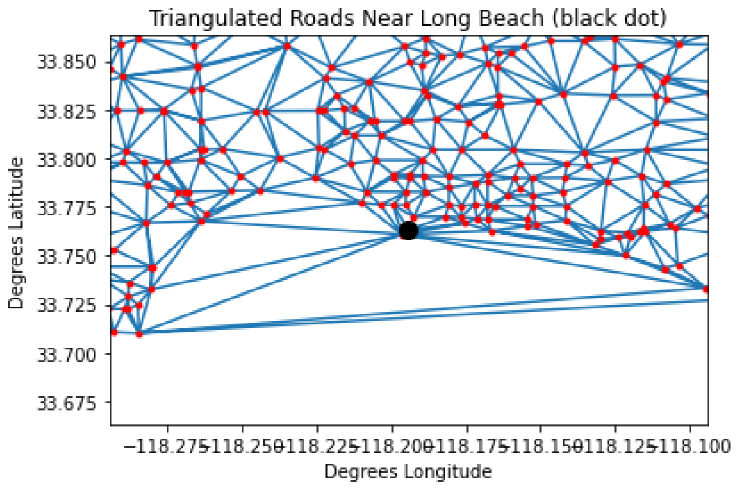

Figure 7: Triangulated roads near Long Beach (represented by black dot)

The picture shows that 5 roads are connecting Malibu and 4 roads leading to Long Beach. Due to the shortboard effect, only 4 roads eventually entered Long Beach. However, we got the result 4. The reason may be some roads are not real, such as the road crosses the Santa Monia bay. We will clear this fake road is Question 14 and try again in Question 15 .

Question 14

$\hat{G}_\Delta$ has been plot on actual coordinates as Figure 8. Now it looks more like "real" Los Angeles. The unreal roads have been removed from some special areas, such as the Topanga State Park in the west, Angeles National Forest in the north, Point Mugu State Park in the northwest, and the most obvious, the coastline of Santa Monia.

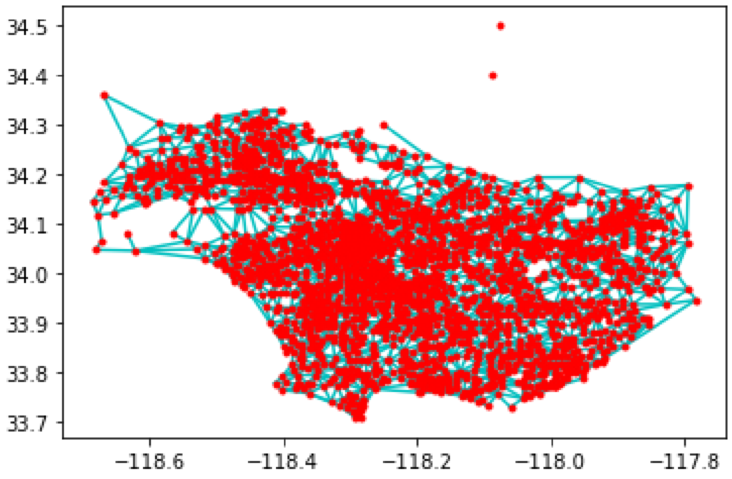

Figure 8: $\hat{G}_\Delta$ on actual coordinates with threshold applied.

Question 15

This time, the number of edge-disjoint paths is the real number connected with Malibu. And because some of the fake roads have been cleared, the maximum flow also decreases a lot, which are shown as Table 3. And the triangulated roads near Malibu and Long beach after pruning are shown in Figure 9 and 10 respectively. Easy to verify, the final calculation results of the disjoint paths in *iGraph* match what we saw on the road map. Now there are only two roads connected with Malibu. And the road near Long Beach is keeping the same. So that the flow of Malibu is the limitation for the maximum flow.

Lane	Maximum flow	Number of edge disjointed path	Aggregate maximum flow
L_1	2896.163	2	6191.658
L_2	3295.494		

Table 3: Max flow and disjoint paths from Malibu to Long Beach with road pruning.

Question 15 continued on next page...

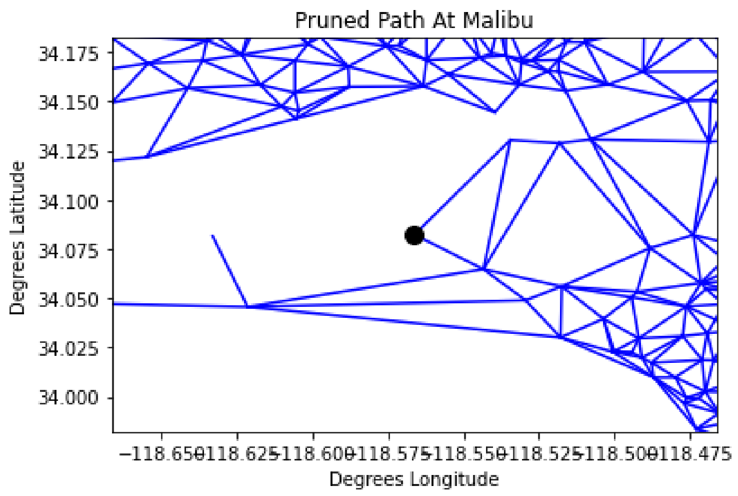

Figure 9: Triangulated roads near Malibu (represented by black dot) after pruning.

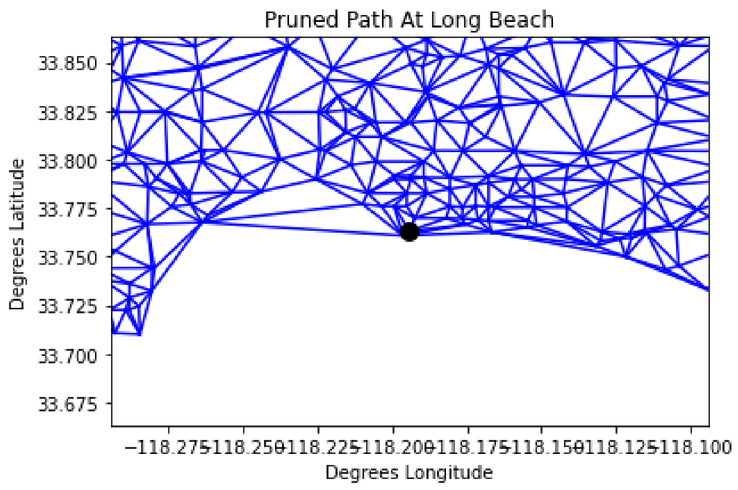

Figure 10: Triangulated roads near Long Beach (represented by black dot) after pruning.

Own Task

In this part, we are going to define and implement some our own tasks other than the tasks already defined in the project handout. In particular, we are going to define some tasks based on the **Uber dataset**, and these tasks are developed based on Question 10, which asks us to plot the trajectory that Santa Clause has to travel within the Los Angeles area based on our approximate solution for the traveling salesman problem (*TSP*). We have defined three tasks and are discussed in the following:
1. Question 10 asks us to plot the trajectory of Santa, and in this question, we just plot the trajectory in Python by using the built-in 'plot' function. This is not interesting because we can only see the trajectory. Thus, the first task we proposed is to map the trajectory we plotted onto the actual map of Los Angeles. By doing so, we can get a better view of Santa's routes and locations that Santa needs to visit. We implemented this idea by using *Tableau*, which is a widely used data visualization software. We believe that as potential data scientists, it is important to learn how to use *Tableau*. And our result is shown as Figure 11.

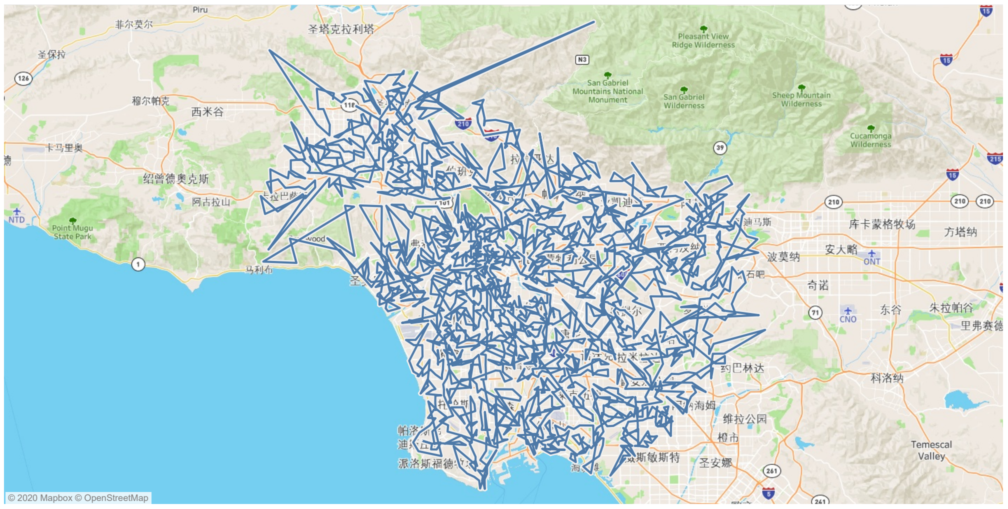

Figure 11: The optimal Santa trajectory for gift delivery mapped onto the actual LA map.

2. Given that we have the trajectory of Santa, the second task we defined is to find out the total distance that Santa has to travel in this *TSP*. We already have a set of coordinates representing Santa 's trajectory, thus we can use this information to calculate the total distance. In particular, we are using the *Haversine* formula to calculate. Detailed information can be found by visiting the Wikipedia page: https://en.wikipedia.org/wiki/Haversine_formula. And according to our calculation, in this TSP, Santa has to travel a total distance of 4364.616 km just within the LA area. Salute to Santa Claus!
3. Next, given the total distance that Santa has to travel, a natural question is that what is Santa's speed. Thus, we defined our third task to be to calculate Santa's travel speed. In terms of the time period, we define that Santa usually travel between 10 p.m. to 4 a.m., which gives us a total time of 6 hours. Thus, the result is 727.436 km/hr, which is almost as fast as a jet plane. This time, we want salute to Santa Claus?s reindeer! Really hope LAPD will not punish Santa for his speeding!