**ECE 232E - Project 5**

**Graph Algorithms**

*Zhechen Xu (805030074), Yufei Hu (404944367)*

*Qi Zeng (204946904), Liangkun Zhao (204947003)*

*June 15, 2018*

# **Part 1: Stock Market**

**Q1:**

According to the definition:



Considering the best and worst situation, the upper bound of it is **1**, while the lower bound of it is **-1**. After analyzing the formulations and investigating some references, there are five reasons we should use log-normalized return instead of regular one:

**Reason 1: Time-additivity.**

**Reason 2: Log-normality:**

if we *assume* that prices are distributed [log normally](http://en.wikipedia.org/wiki/Log-normal_distribution) (which, in practice, may or may not be true for any given price series), then  is conveniently [normally distributed](http://en.wikipedia.org/wiki/Normal_distribution), because:



**Reason 3: Approximate raw-log equality:**

when returns are very small (common for trades with short holding durations), the following approximation ensures they are close in value to raw returns:



**Reason 4: Mathematical ease:**

from [calculus](http://en.wikipedia.org/wiki/Calculus), we are reminded (ignoring the [constant of integration](http://en.wikipedia.org/wiki/Constant_of_integration)):



This identity is tremendously useful, as much of financial mathematics is built upon [continuous time stochastic processes](http://en.wikipedia.org/wiki/Continuous-time_stochastic_process) which rely heavily upon integration and differentiation.

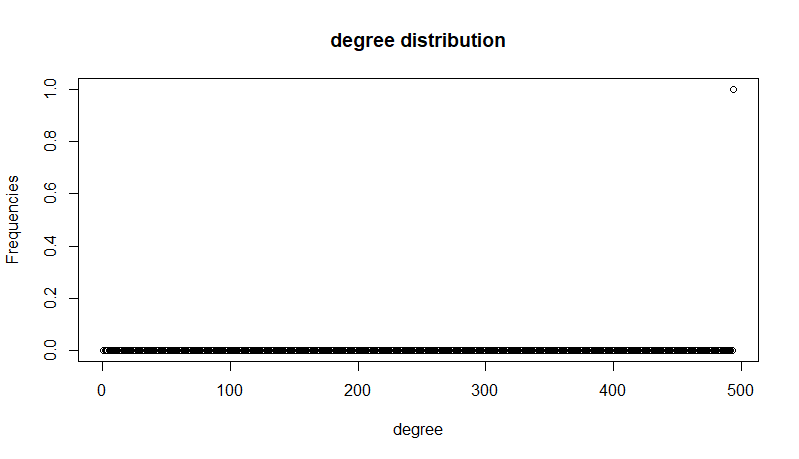
**Reason 5: Numerical stability:**

Addition of small numbers is numerically safe, while multiplying small numbers is not as it is subject to [arithmetic underflow](http://en.wikipedia.org/wiki/Arithmetic_underflow). For many interesting problems, this is a serious potential problem. To solve this, either the algorithm must be modified to be numerically robust or it can be transformed into a numerically safe summation via logs.

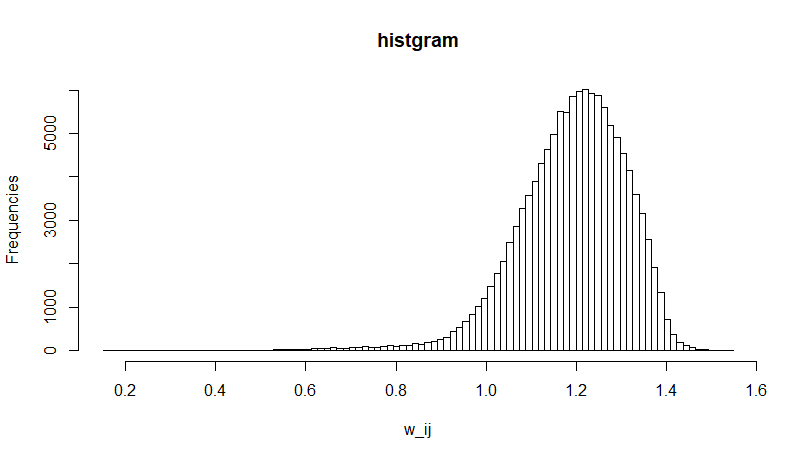
*Reference: https://quantivity.wordpress.com/2011/02/21/why-log-returns/*

**Q2:**

Degree distribution of the correlation graph:

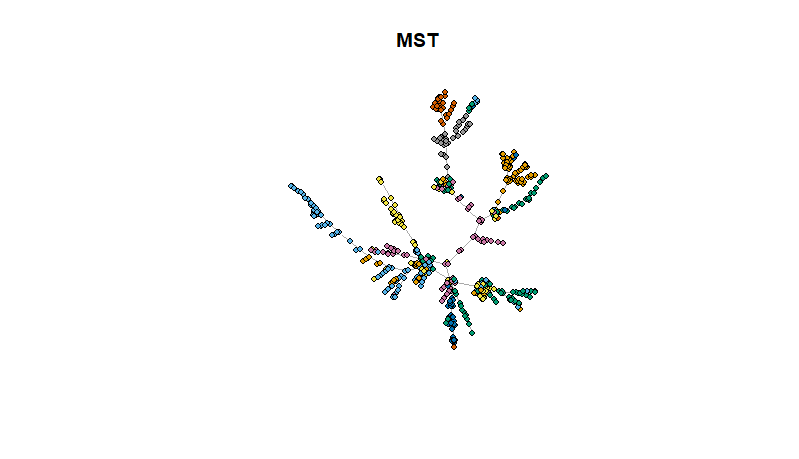


It’s a one bar plot. The un-normalized distribution of edge weights:



**Q3:**

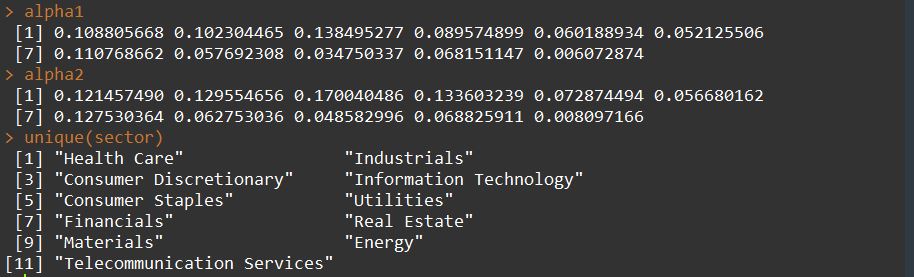
The MST and color-coded nodes:



As we can see in the plot, those nodes with the same color have more possibility to get together. This structure is called Vine clusters, which, in our problem, reflects the stocks in the same kind of sector.

**Q4:**

The values of alpha for the two cases are summarized in the following table:



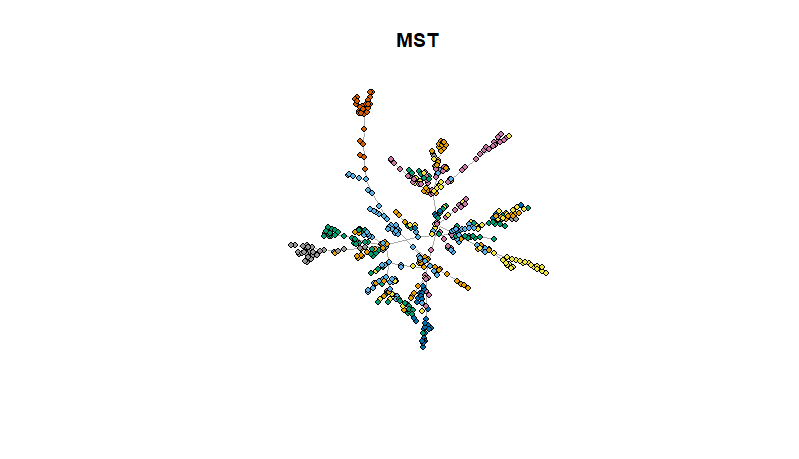
|  |  |  |
| --- | --- | --- |
| **Sectors** | **Alpha 1** | **Alpha 2** |
| Health Care | 0.108805668 | 0.121457490 |
| Industrials | 0.102304465 | 0.129554656 |
| Consumer Discretionary | 0.138495277 | 0.170040486 |
| Information Technology | 0.089574899 | 0.133603239 |
| Consumer Staples | 0.060188934 | 0.072874494 |
| Utilities | 0.052125506 | 0.056680162 |
| Financials | 0.110768662 | 0.127530364 |
| Real Estate | 0.057692308 | 0.062753036 |
| Materials | 0.034750337 | 0.048582996 |
| Energy | 0.068151147 | 0.068825911 |
| Telecommunication services | 0.006072874 | 0.008097166 |

The values are calculated from the cleaned data (494 stocks). We first conducted data preprocessing and discarded 11 stocks with different time length.

As shown in the table, the values given by the second definition are always larger than the one given by the first definition. The reason behind that is because instead of analyzing one’s neighbors, we just defined the probability as the |Si|/|V|, which is clearly not a very good formulation when we have the real data. If we use the first definition, when calculating alpha, not all the stocks belong to one single sector have relationship with the one we investigated. And that’s why the second one is always larger than the first one.

**Q5:**

The MST given by the weekly data:

****

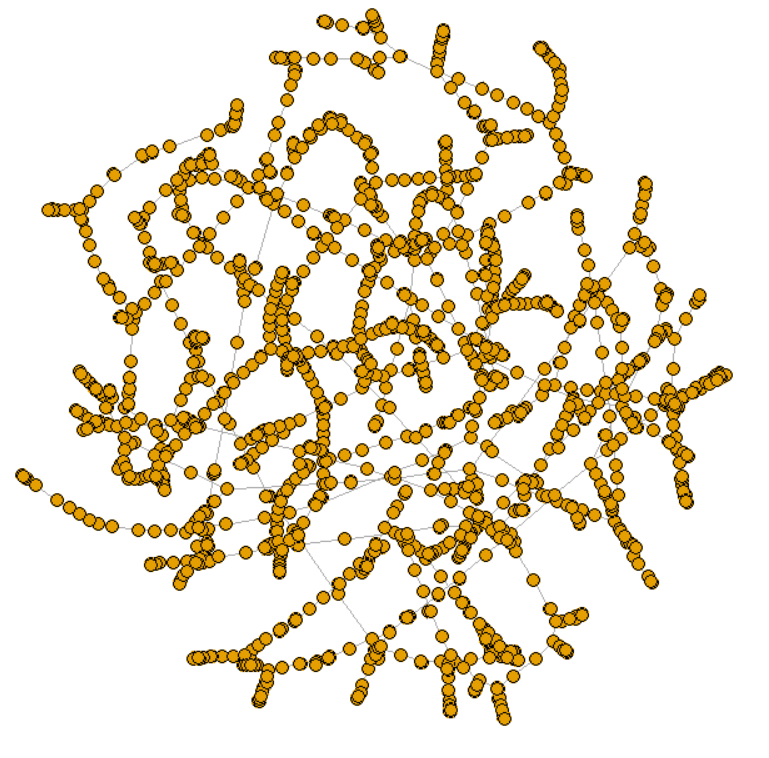
Comparing with the plot in question 3, the stocks with same color do not stick together as a cluster. Rather, they are relatively distributed in the all graph. This is because when generating MST in this question, we used every Monday data, in other words, we discarded 4 day’s data when using every Monday data so that the result we got is very rough comparing with the accurate one computed using all the weekday’s data.

# **Part 2: Let’s Help Santa!**

**Q6:**

The cleaned graph has **1880** nodes and **311802** edges.

**Q7:**



I have randomly sampled 6 source-destination pairs which are shown in the table below:

|  |  |  |
| --- | --- | --- |
| **Source Street** | **Destination Street** | **Traveling Time (seconds)** |
| 200 Genoa Court, Vallejo | 200 Kirkland Ranch Road, American Canyon | 230.63 |
| 3500 Brooks Avenue, Santa Rosa | 200 Firethorn Drive, Rohnert Park | 228.16 |
| 400 Monti Circle, Pleasant Hill | 0 Bartlett Court, Pleasant Hill | 243.39 |
| Ed Taylor Trail, Burlingame Hills, Burlingame | 1200 Manzanita Drive, Millbrae | 74.34 |
| 1400 Essex Way, West San Jose, San Jose | 4000 Moreland Way, West San Jose, San Jose | 148.95 |
| 400 East Tabor Avenue, Fairfield | 2000 Thrush Way, Fairfield | 120.57 |

All these pairs quite make sense. I have manually checked some of them on Google Map. For example, for the pair 1400 Essex Way, West San Jose, San Jose to 4000 Moreland Way, West San Jose, San Jose, the recorded mean traveling time is around 2.5 minutes while Google Map suggests 3 minutes.

**Q8:**

After I randomly sampled 1000 distinct triangles and around **94%** of them satisfies triangle inequality.

**Q9:**

In order to derive the approximate TSP cost, I have run DFS algorithm on the MST we have just derived in Question 7. The starting node is picked randomly. I will then get a trajectory. The approximate TSP cost is derived by summing all the weights in the derived trajectory. To those vertices pair that do not have a path between them, I have estimated their weights by computing the ratio between the sum of all weights in the original graph and the sum of Euclidean distances of those corresponding edges and use this ratio to convert the Euclidean distance of the vertices pair into their estimated weight. Then, the optimal TSP cost is derived simply by summing all the weights in the MST we got. The final result is shown below:



**Q10:**



The first 10 streets are shown below:

*200 Walnut Street, Presidio Heights, San Francisco*

*700 Presidio Avenue, Western Addition, San Francisco*

*2500 Turk Street, Richmond District, San Francisco*

*0 Rossi Avenue, Richmond District, San Francisco*

*600 3rd Avenue, Richmond District, San Francisco*

*200 3rd Avenue, Richmond District, San Francisco*

*800 Balboa Street, Richmond District, San Francisco*

*1500 Balboa Street, Richmond District, San Francisco*

*200 16th Avenue, Richmond District, San Francisco*

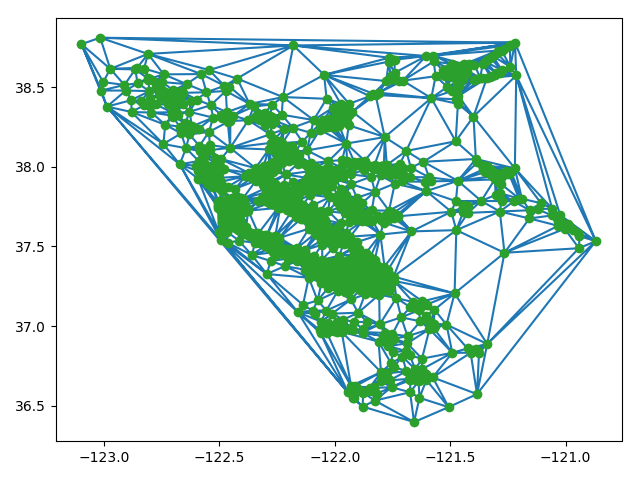
*200 21st Avenue, Central Richmond, San Francisco*

Similar to thing I did in Question 7, I have once again manually checked some of the streets in Google Map. It turns out that all of them are quite close to each other, thus indicating that the trajectory we derived are quite intuitive.

# **Part 3: Analyzing the Traffic Flow**

**Q11:**

After applying Delaunay triangulation algorithm to the graph in part2, we can plot the graph with given vertices and edges. The result is below:



According to Wikipedia, Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid sliver triangles.

**Q12:**

We calculate the car flow using the hints and store the car flow results to each of the edge in G∆.

**Q13.**

The maximum number of cars that can commute per hour from Stanford to is 15132.36. The number of edge-disjoint paths between the two spots is 8. Because the nodes are too crowded between (-122.175982, 37.429686) and (-122.064562, 36.974169) coordinates, we cannot check our result using the graph in Q11.

**Q14.**

After applying a threshold on the travel time of the roads in G∆ to remove the fake edges we get the modified graph as below:

In the graph Golden Gate Bridge and Dambarton Bridge are preserved.

**Q15.**

In the new graph G∆, the preservation rate is 100%. Because we remove the fake edges, the remaining nodes with edges should preserve triangle inequality.