EE232E Graphs and Network Flows

Spring 2017_Homework_4

Team Member:
Xiongfeng Hu (304753117)
Younan Liang (504759929)
Dui Lin (504759948)

1. Problem Solutions

Problem 1. Calculating Correlations among Time Series Data

In this problem, we calculate the cross correlation coefficient of two different stock-return time series by following equation:

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

where i and j denote two stocks and $r_i(t) = logp_i(t) - logp_i(t-\tau)$. Here we use the closing price for pi(t) and we call ri(t) as log return.

Several benefits of using log returns, both theoretic and algorithmic are as below:

1) Log-normality:

if we assume that prices are distributed log normally (which, in practice, may or may not be true for any given price series), then $r_i(t)$ is conveniently normally distributed, because:

$$r_i(t) = logp_i(t) - logp_i(t-\tau) = log\frac{p_i(t)}{p_i(t-\tau)}$$

This is handy given much of classic statistics presumes normality.

2) 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:

$$r_i(t) = log\frac{p_i(t)}{p_i(t-\tau)} = log(1 + \frac{p_i(t)-p_i(t-\tau)}{p_i(t-\tau)}) \approx \frac{p_i(t)-p_i(t-\tau)}{p_i(t-\tau)}, \frac{p_i(t)-p_i(t-\tau)}{p_i(t-\tau)} \ll 1$$

3) Time-additivity:

consider an ordered sequence of n trades. A statistic frequently calculated from this sequence is the compounding return, which is the running return of this sequence of trades over time:

$$\frac{p_1(t)}{p_1(t-\tau)} \cdot \frac{p_2(t)}{p_2(t-\tau)} \cdot \ldots \cdot \frac{p_n(t)}{p_n(t-\tau)} = \prod_i \frac{p_i(t)}{p_i(t-\tau)}$$

This formula is fairly unpleasant, as probability theory reminds us the product of normally-distributed variables is not normal. Instead, the sum of normally-distributed variables is normal (important technicality: only when all variables are uncorrelated), which is useful when we recall the following logarithmic identity:

$$r_i(t) = logp_i(t) - logp_i(t-\tau) = log\frac{p_i(t)}{p_i(t-\tau)}$$

Thus, compounding returns are normally distributed. Finally, this identity leads us to a pleasant algorithmic benefit; a simple formula for calculating compound returns:

$$\sum_i r_i(t) = r_1(t) + r_2(t) + \ldots + r_n(t) = logp_n(t) - logp_0(t)$$

Thus, the compound return over n periods is merely the difference in log between initial and final periods. In terms of algorithmic complexity, this simplification reduces O(n) multiplications to O(1) additions. This is a huge win for moderate to large n. Further, this sum is useful for cases in which returns diverge from normal, as the central limit theorem reminds us that the sample average of this sum will converge to normality (presuming finite first and second moments).

4) Mathematical ease:

from calculus, we are reminded (ignoring the constant of integration):

$$e^x = \int e^x dx = \frac{d}{dx} e^x = e^x$$

This identity is tremendously useful, as much of financial mathematics is built upon continuous time stochastic processes which rely heavily upon integration and differentiation.

5) Numerical stability:

addition of small numbers is numerically safe, while multiplying small numbers is not as it is subject to 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.

Problem 2. Constructing Correlation Graphs

In this problem, we construct correlation graphs by viewing every stock as a node in the graph. And we define the weight of edge ij as $d_{ij} = \sqrt{2(1 - \rho_{ij})}$, where ρ_{ij} represents the cross correlation coefficient of two different stock-return time series as described in problem one and we set time scale $\tau = 1$. Then we construct a weighted graph G with adjacency matrix D = [dij]. The number of nodes in the graph is 505 and the number of edges is 127260.

Now we Plot the histogram of dij's.. From the histogram, we can see that the dij is distributed normally. The mean is around 1.2 and the variance is 0.01589919. This result satisfied our presumed normality in problem one. The plot is as below:

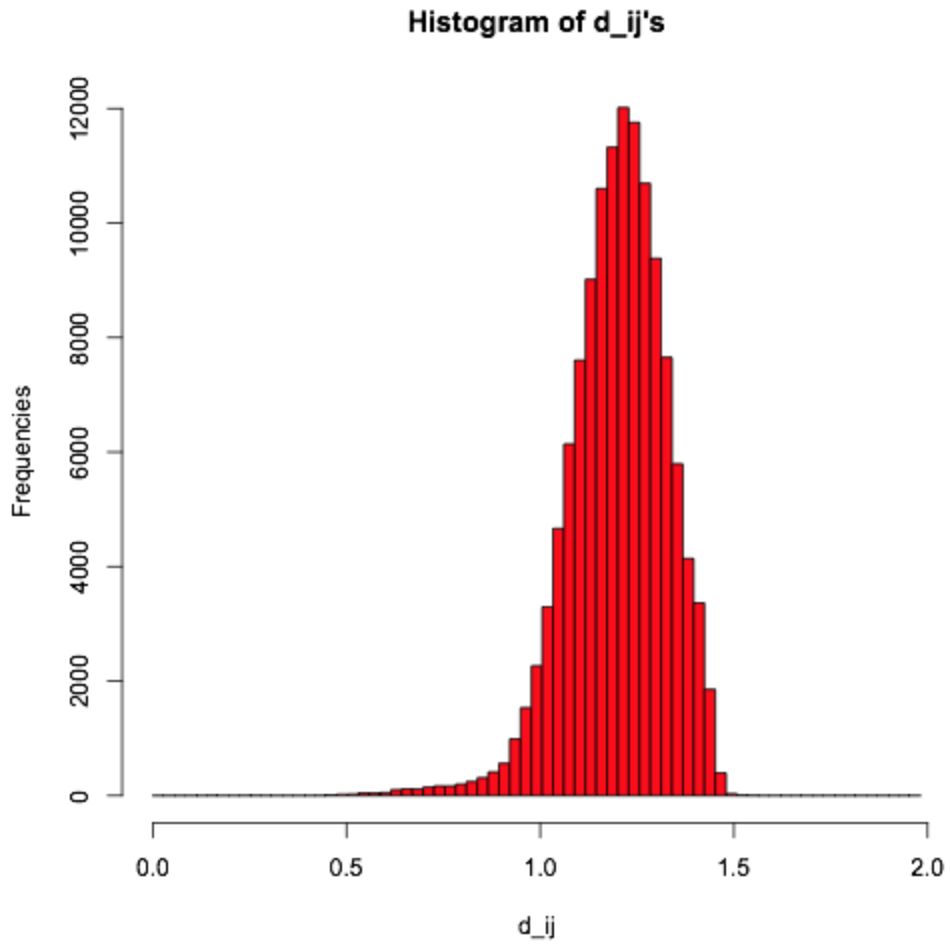

Figure 2.1 Histogram of dij

Problem 3. Finding Minimum Spanning Trees (MSTs) for the Correlation Graphs

Minimum spanning tree. An edge-weighted graph is a graph where we associate weights with each edge. A minimum spanning tree (MST) of an edge-weighted graph is a spanning tree whose weight (the sum of the weights of its edges) is no larger than the weight of any other spanning tree.

In this problem, we first compute the MST of the correlation graph using R function mst(g1, weights = P.data$weights). Then we plot the MST and color-code the nodes based on sectors. In the MST network, number of nodes is 505 and number of edges is 504 which satisfies the properties of tree graph.

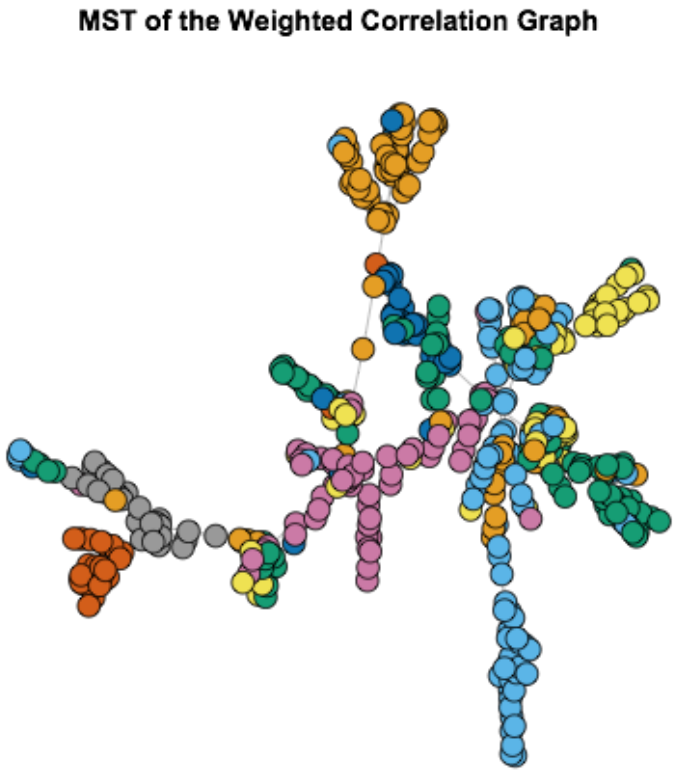

Figure 3.1 MST of the Weighted Correlation Graph

As time scale is set as $\tau = 1$, we can see from the plot that nodes of the same color tend to stick together. This pattern tells us that MST for the Correlation Graphs construct a vine-cluster structure, where the nodes representing the stocks in the same sector are clustered together.

Problem 4. Evaluating Sector Clustering in MSTs

In this question, we want to predict the market sector of an unknown stock based on the MST that we just found. One might be interested in doing so just based on the immediate neighbors of a stock in the MST. Evaluate the performance of such a method as follows:

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

The following table shows some cases where each MST node is assigned a random sector. That is, the probability a node is assigned the sector Si is (# of stocks with label Si)/Total # of stocks.

Sector Name	Alpha	Random Sector
Health Care	0.8875000	0.118811881
Industrials	0.7723857	0.128712871
Consumer Discretionary	0.8049020	0.168316832
Information Technology	0.6750000	0.138613861
Consumer Staples	0.8036036	0.073267327
Utilities	0.9196429	0.055445545
Financials	0.8290867	0.130693069
Real Estate	0.9005376	0.061386139
Material	0.6866667	0.049504950
Energy	0.9901961	0.067326733
Telecommunication Services	0.7500000	0.007920792

Table 4.1. Comparisons between alpha and random sector

Also, from the table 4.1, we can plot the function of Alpha vs Random Sector, as shown in figure 4.2. We find that the alpha value and random sectors fluctuate, but not fluctuate very much.

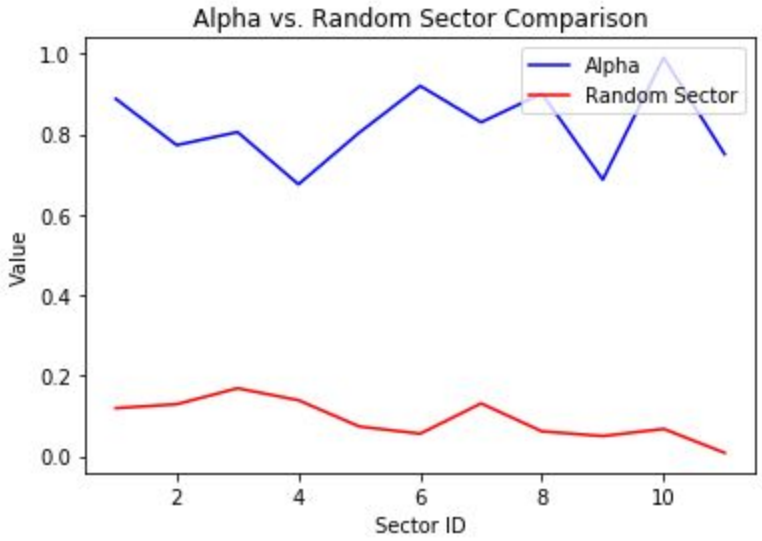

Figure 4.2 Alpha vs Random Sector Comparison

Problem 5. Δ-Traveling Salesman Problem

We can prove that the triangle inequality holds for the fully connected graph G by two ways: The first way is that we can Python three loops and check all the triangles in the graph G; the second way is to mathematically prove it: In a fully connected graph G, Suppose d(x,y)+d(y,z) < d(x,z). By definition, d(x,y) is the length of the shortest path from x to y, and d(y,z) is the length of the shortest path from y to z. However, d(x,z) is the length of the shortest path from x to z, so d(x,y)+d(y,z)can't be less than d(x,z). So we have a contradiction. And this is the mathematical proof of this statement.

Then we use Python Euler-Tour to find an approximate tour for the TSP. The results are shown as follows:

```
> cat ("Sum of weights of min spanning tree: ",lower_bound,"\n")
Sum of weights of min spanning tree:  432.2612

> cat ("Sum of weights of double min spanning tree: ",upper_bound,"\n")
Sum of weights of double min spanning tree:  864.5223

> cat ("Sum of edge weights of TSP solution: ", tsp_weight,"\n")
```

Sum of edge weights of TSP solution: 432.2612

However, we cannot give a guarantee on the globally optimality of our solution. This is because w (MST) < w (TSP) < 2w (MST). MST, instead of TSP, is the lower bound of the best solution.

Bonus: We also try to use external package to solve TSP problem, for example, we use fast insertion algorithm in package "TSP" (https://cran.r-project.org/web/packages/TSP/TSP.pdf) to calculate TSP weight sum, and the estimated result is 466.95, which is worse than the previous solution.

Problem 6. Constructing Correlation Graphs for weekly data

In the previous questions, we used daily closing prices for stocks to compute returns. In this question, we sample the stock data weekly on Mondays. Then we plotted the MST using only the data on Monday. The MST graph is shown as figure 6.1:

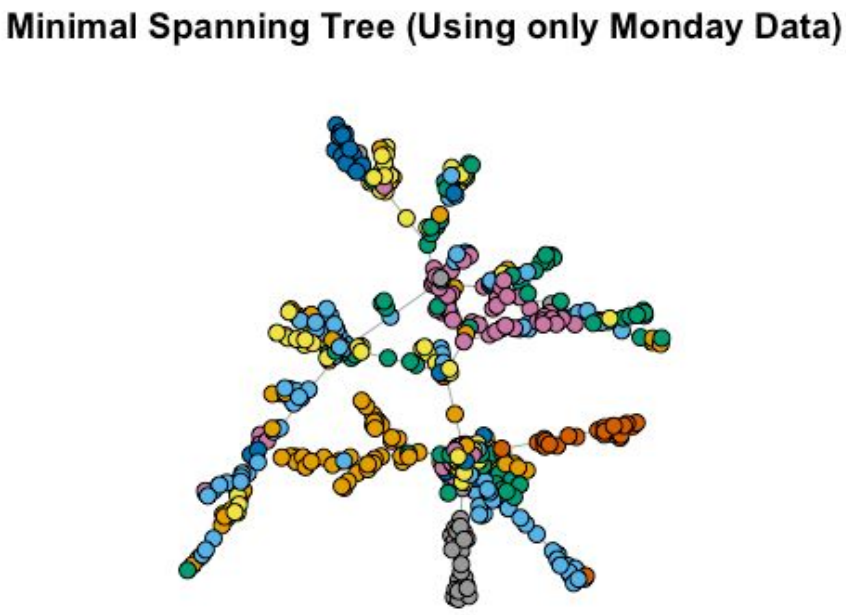

Figure 6.1 MST Tree (Using only Monday Data)

Then we compare the two results: based on daily and weekly stock prices. Here we ignore the holidays on Mondays, since there is no data available. The MST of weekly price is shown as question 3, figure 3.1; while the MST of Monday price is shown as figure 6.1. Compare with two MST graphs. In the MST of weekly price, it is very clear that nodes of the same color tend to stick together. This pattern tells us that MST for the Correlation Graphs construct a vine-cluster structure, where the nodes representing the stocks in the same sector are clustered together. However, in the MST of Monday price, this vine-cluster structure pattern is not very clear. The nodes with the same color seem to be somewhat distributed through the MST graph. This is to say that the correlation is not so close between stocks in the same sector. And this result is intuitively right. Instead of $\tau = 1$ in the weekly data, $\tau = 5$ (trading days) in the Monday data, which results in more fluctuation, unpredictable price change of the stock, and less correlation than the weekly data as shown in problem 3.

However, in this problem, there is a problem here, we cannot plot histogram here, because if Monday is holiday, the data will be omitted.

Problem 7. Modifying Correlations

First, we plot the histogram of ρij ’s from daily data. The figure is shown as follows:

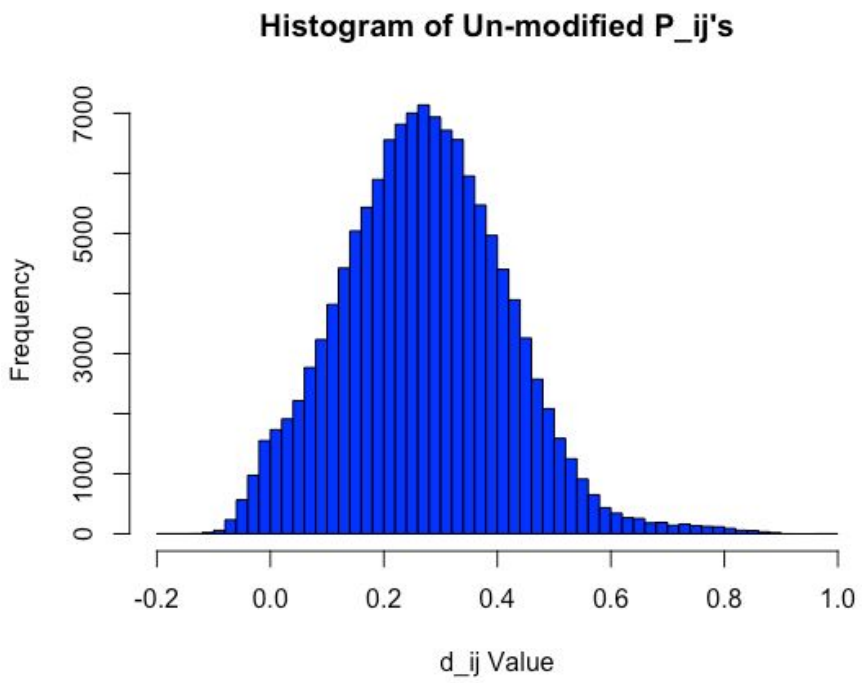

Figure 7.1 Histogram of ρij ’s from daily data

From figure 7.1, we can see that the correlation factors are centered at 0.3. This means that the stock prices in the whole market are somewhat positively related. This result is intuitively right. For example, when the economic environment is good, most of the company will earn lots of profits, which results in the stock index increase. And vise versa for the bad economic environments.

Then, set all ρij ’s larger than 0.3 as -1; for ρij ’s ≤ 0.3, keep the original value. Calculate dij = Sqrt (2(1 −ρij)). The results are shown as follows:

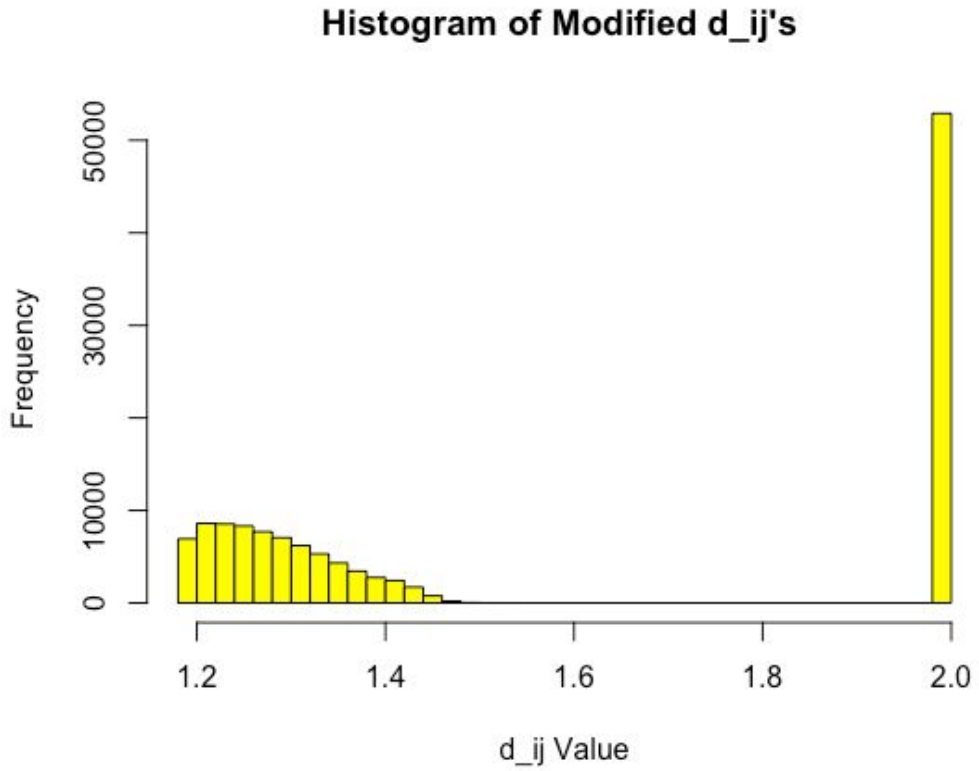

Figure 7.2 Modified Histogram of ρij ’s from daily data

From figure 7.2, we can find that the modified correlations have two centered distributions: 1.3 and 2.0. The results are obvious, since that the all ρij ’s larger than 0.3 are set as -1, and corresponds to dij=2.0. The remaining ρij ’s smaller than 0.3 is unchanged, so the modified dij will be slightly larger than 1.0.

Then we construct the graph and run MST. The results are shown as figure 7.3:

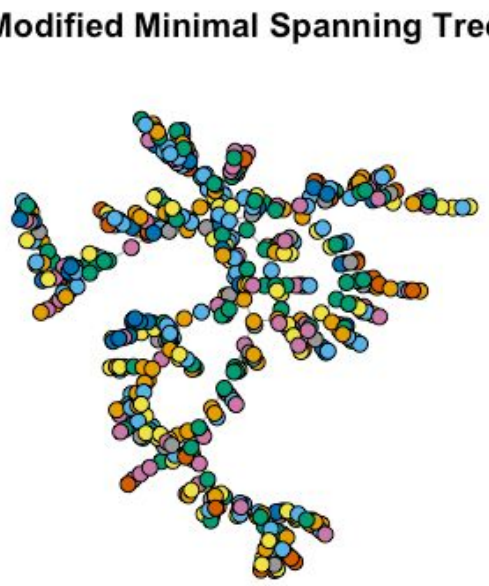

Figure 7.3 Modified Minimal Spanning Tree (MST)

In figure 7.3, the nodes with the same color are totally randomly distributed through the whole MST graph. So the vine cluster patterns no longer exist with modified correlations. This is because the correlations between the nodes are no longer positive, as shown in figure 3.1 and figure 6.1.

Problem 8. Evaluating Sector Clustering in MSTs

In this problem, we come up with a generative model for vine cluster graphs. First we begin with 2 nodes and attach a tag “1”. Whenever we generate a new node, we attach the new node with the existing nodes with the probability to the value of their tags. When this attachment is completed, the attach of this new node is attached a tag “1”, and tag for the existing nodes are changed to 1/deg(this existing node).

For example, when we generate the third node into the graph, this node is randomly connected to one of the starting nodes. Then this new node is attached with a tag “1”, and the neighbor of this new node is changed to 0.5. Following this process again and again until the number of nodes is large enough in the network.

We find that the graph created by generative looks similar compared to figure 3.1 and figure 7.3. Therefore, we see the vine cluster pattern again. However, we cannot guarantee that every time we generate a random graph, there must be a vine cluster pattern.

2. Useful Links

[1]. https://math.stackexchange.com/questions/737487/proof-for-a-graph-distance
[2]. https://drive.google.com/drive/folders/0B2I9Ux-1OJOmRGZ6c2hoRTVYWjQ?usp=sharing
[3]. https://www.dropbox.com/s/83l60htndqpn3fv/finance_data.zip?dl=0
[4]. https://github.com/feynmanliang/Euler-Tour