

# EE232E HW4 Report

TEAM MEMBERS: Qinxin Li 704774783

Jiatong Chen 404361029

Yuliang Fang 404759010

## 1. Calculating Correlations among Time Series Data

By definition, the cross correlation coefficient of 2 different stock-return time series is  $\rho_{ij} = \text{cor}(r_i, r_j)$  where  $r_i = \log p_i(t) - \log p_i(t-\tau)$  are log returns of stock price. Here we use the closing price for  $p_i(t)$  and we set  $\tau=1$ . We use python to read the csv files of stock data in finance\_data/data directory and we use `numpy.diff`, `numpy.log` and `numpy.corrcoef` to get the correlation coefficient of the stock prices.

We use log return to measure the change of prices between days. There are mainly 5 reasons for using the log return.

First, 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  $\log(1 + r_i)$  is conveniently normally distributed, because:

$$1+r_i = \frac{p_i}{p_j} = \exp^{\log \frac{p_i}{p_j}}$$

This is handy given much of classic statistics presumes normality.

Second, 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:

$$\log(1 + r_i) \approx r_i \text{ when } r_i \approx 0$$

Third, 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:

$$(1 + r_1)(1 + r_2)(1 + r_3)\dots = \prod (1 + r_i)$$

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:

$$\sum \log(1 + r_i) = \log(1 + r_1) + \log(1 + r_2) + \log(1 + r_3) + \dots = \log(p_n) - \log(p_0)$$

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).

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

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

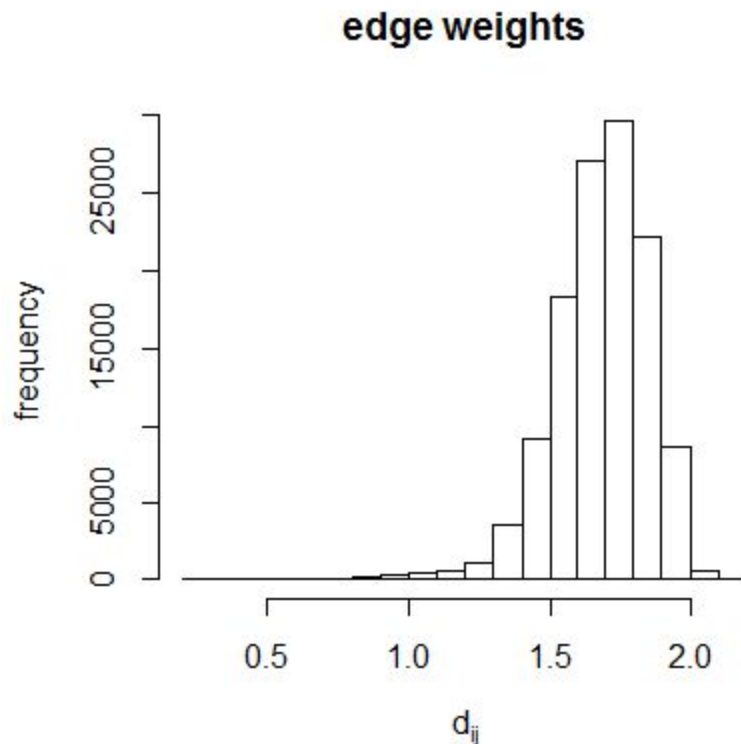
This identity is tremendously useful, as much of financial mathematics is built upon continuous time stochastic processes which rely heavily upon integration and differentiation.

Fifth, 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.

## 2. Constructing Correlation Graphs

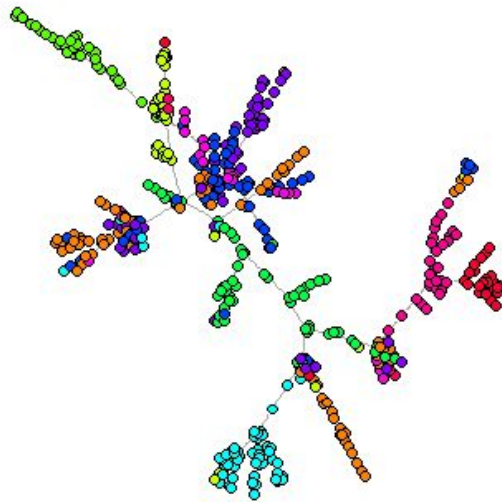
We can construct a graph for the stocks where each stock is a node. The weight matrix of the graph is calculated as  $d_{ij} = \sqrt{2(1 - \rho_{ij})}$ . We use `igraph.Graph.Adjacency` to construct the graph. The histogram of  $d_{ij}$ s are shown as below:



We can see that the edge weights are mainly distributed in [1.5,2.0] range.

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

When we get the graph of stocks, we can run minimum spanning tree algorithms on the graph. Actually, because it is a tree, the number of edges is always number of vertices minus 1. There are totally 494 nodes in the minimum spanning tree. We color the nodes belonging to each cluster and plot the minimum spanning tree to observe its structure. The plotted graph is shown below.



We can see that the stocks in the same sector are clustering. It means that the length of edges does show the closeness between nodes, and stocks in the same sector belongs to the same class.

### 4. Evaluating Sector Clustering in MSTs

We want to predict the market sector of an unknown stock based on the MST just found. The metrics of the performance evaluation is as follows:

$$\alpha = \frac{1}{|V|} \sum_V P_i \in S_i$$

Which means  $\alpha$  is just the proportion of immediate neighbors of a node  $v_i$  which belongs to the same sector as  $v_i$ .

We calculate  $\alpha$  based on the MST we have created and by assigning sectors randomly on the nodes by the probability  $p_i$  (but keep the percentile of each sector not changed). We then calculated the two  $\alpha$ .

The  $\alpha$  calculated on the original MST is 0.6887767

The  $\alpha$  calculated on the randomly assigned MST is 0.1272154

The former  $\alpha$  is greater than the latter one. It means that the clustering is good on the original MST.

## 5. $\Delta$ -Traveling Salesman Problem

The triangle inequality property can be proved by checking whether for each  $i, j$ , We have  $d(i, j) < d(i, k) + d(j, k)$  for an arbitrary  $k$ .

The TSP solution can be solved by the following steps:

1. Duplicate the MST graph edges and find an Euler path in the new graph.
2. Short-cutting the Euler path to get the optimal result.

The Euler path is actually an 1-approximation solution of the TSP problem. Since the MST can not be traversed by an Euler path, the TSP must be longer than the total edge length of the MST. But the Euler path is no longer than twice of the total MST edge length, while the optimal TSP solution is no longer than the Euler path. Then  $TSP < Euler\text{-}path < 2 * TSP$ , which proves our conclusion.

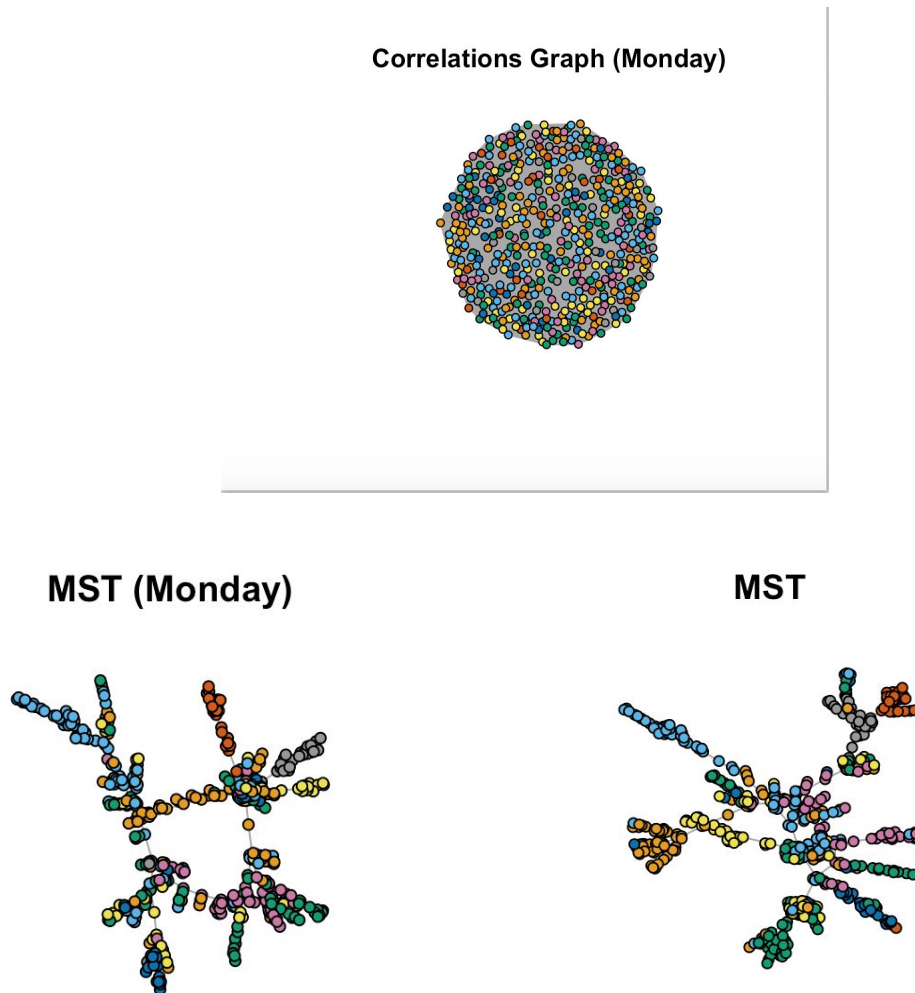
In our code, we use fleury algorithm to find the Euler path. And we do the short-cutting by deleting the revisited vertices recursively until no vertex is deleted. Then the path is the global optimal solution of the TSP problem. One of the solution is:

[4, 36, 40, 33, 38, 44, 46, 73, 252, 35, 64, 72, 88, 111, 106, 133, 261, 281, 137, 257, 299, 320, 253, 317, 41, 129, 224, 229, 288, 376, 101, 81, 428, 256, 20, 62, 447, 477, 74, 22, 273, 419, 379, 339, 407, 399, 14, 11, 18, 26, 24, 305, 23, 31, 43, 50, 105, 84, 70, 60, 211, 234, 280, 283, 30, 34, 342, 485, 15, 127, 100, 147, 193, 346, 67, 425, 148, 155, 125, 1, 143, 170, 146, 158, 198, 251, 182, 230, 165, 255, 201, 319, 440, 369, 410, 473, 452, 3, 2, 5, 295, 55, 118, 348, 302, 277, 221, 65, 208, 239, 260, 262, 219, 347, 471, 327, 478, 450, 162, 205, 160, 128, 214, 409, 97, 27, 210, 166, 132, 232, 290, 400, 233, 279, 282, 225, 338, 341, 390, 439, 104, 276, 345, 366, 488, 164, 131, 12, 303, 130, 184, 259, 68, 57, 66, 80, 99, 39, 136, 83, 87, 109, 92, 117, 178, 300, 378, 362, 289, 411, 430, 460, 476, 456, 365, 207, 265, 306, 309, 377, 474, 417, 403, 384, 387, 443, 9, 484, 335, 451, 470, 493, 408, 482, 435, 490, 370, 489, 364, 462, 486, 475, 427, 49, 58, 270, 215, 271, 388, 392, 453, 48, 168, 190, 240, 202, 172, 102, 171, 179, 153, 42, 17, 96, 116, 216, 243, 332, 167, 21, 19, 51, 61, 112, 114, 492, 448, 357, 349, 367, 436, 212, 310, 389, 142, 402, 418, 173, 124, 141, 174, 139, 209, 352, 220, 304, 331, 437, 441, 191, 181, 189, 152, 199, 204, 360, 433, 444, 415, 446, 297, 373, 247, 337, 469, 315, 438, 465, 371, 296, 254, 274, 278, 420, 354, 313, 250, 32, 37, 29, 47, 248, 314, 293, 395, 432, 0, 404, 321, 156, 356, 449, 238, 350, 454, 393, 312, 416, 330, 468, 423, 56, 69, 82, 77, 95, 121, 85, 107, 16, 103, 119, 94, 135, 115, 91, 76, 93, 138, 52, 194, 245, 217, 244, 222, 263, 266, 213, 227, 322, 406, 325, 374, 459, 463, 481, 382, 461, 368, 294, 422, 457, 405, 464, 59, 78, 86, 71, 90, 120, 149, 237, 268, 291, 108, 195, 163, 183, 223, 140, 264, 329, 414, 483, 267, 333, 287, 323, 480, 326, 359, 375, 122, 298, 334, 394, 396, 424, 336, 177, 269, 75, 113, 226, 200, 45, 53, 154, 391, 159, 25, 272, 340, 292, 328, 351, 380, 301, 241, 8, 386, 429, 487, 343, 344, 467, 372, 458, 434, 6, 161, 203, 218, 383, 235, 397, 324, 385, 413, 381, 466, 472, 426, 412, 401, 144, 123, 98, 176, 355, 54, 284, 180, 187, 151, 197, 145, 157, 126, 196, 285, 169, 134, 442, 353, 455, 249, 311, 398, 275, 431, 192, 185, 7, 13, 188, 236, 231, 242, 307, 246, 110, 150, 175, 186, 28, 363, 479, 421, 358, 10, 286, 228, 316, 318, 491, 445, 63, 79, 206, 258, 361, 308, 89]

## 6. Constructing Correlation Graphs for Weekly Data

For this problem, we compute returns using only data on Monday to sample the stock data. We used the same method as the first part, except that during the data processing, we extract only the data on Mondays. Then, with the same code, we calculate the  $P_{ij}$  of the data, then produce the correlation graph and the MST.

The result correlation graph and MST plot are shown below:



The left plot is the MST plot is the MST using the data only on Monday, the right one is the MST plot using daily data. Different colors represent different sectors. We can see from the two plots that: calculating MST using daily data will produce a MST with more accumulative sectors. From the plot, it's obvious that the plot on

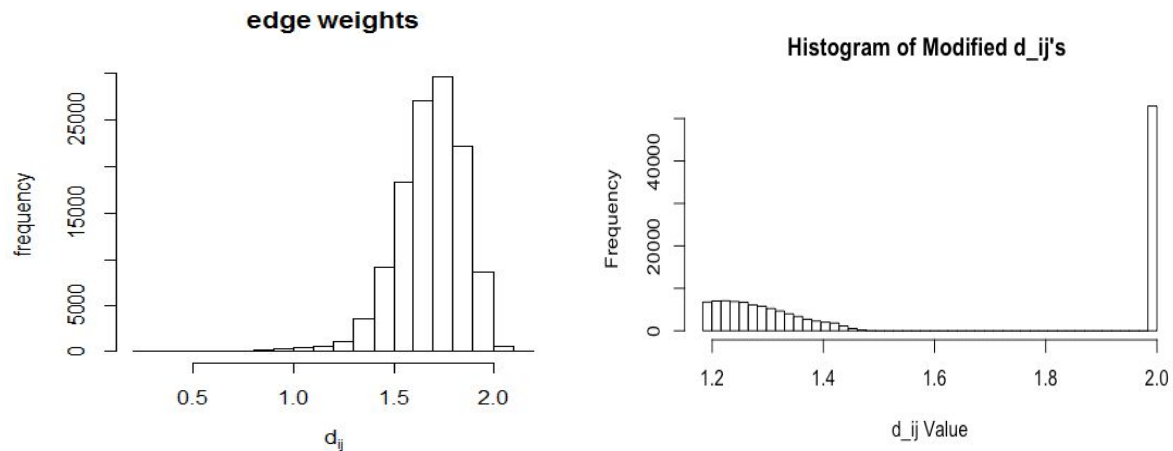
the right has more clear color edges, at the same time, the plot on the left seems to merge every color. It means that sampling the stock data with daily data will have more clear sectors in each branches, which is more accurate than using weekly data.

## 7. Modifying Correlations

For this problem, we keep using the daily data to sample the stock data.

One thing different is that, when we are calculating each  $P_{ij}$ , if the current  $P_{ij}$  value  $> 0.3$ , then we replace the  $P_{ij}$  value with -1. For  $P_{ij}$  values  $\geq 0.3$ , we keep the original value. After this step, we do the same thing as before: calculate  $D_{ij}$  using formula  $D_{ij} = \sqrt{2(1 - P_{ij})}$  and calculate the MST. Also, for this problem, we produce the histograms of  $P_{ij}$ 's from daily data using both modified correlations and unmodified correlations.


The result histograms are shown as below:



The left one is the histogram of the unmodified correlations, and the right one is the histogram of the modified correlations.

The MST graph we get for both the modified and unmodified correlations are shown below:





The idea of this problem is actually based on the Pearson Correlation Coefficient ( $P_{ij}$ ) and Pearson's distance ( $D_{ij}$ ). By definition, Pearson Correlation Coefficient is a measure of the linear correlation between two variables  $X$  and  $Y$ . It has a value between  $+1$  and  $-1$ , where  $1$  is total positive linear correlation,  $0$  is no linear correlation, and  $-1$  is total negative linear correlation, and Pearson's distance is a correlation distance based on Pearson's product-momentum correlation coefficient of the two sample vectors. Since the correlation coefficient falls between  $[-1, 1]$ , the Pearson distance lies in  $[0, 2]$  and measures the linear relationship between the two vectors.

In this problem, we replace all the  $P_{ij}$  values larger than 0.3 with -1, which means we replace all the strong positive correlation with total negative linear correlations. Also, the original Pearson's Distance is changed from a value close to 0 with a value of nearly 2. We know that MST is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. The MST is produced from the Pearson's Distance. So the original edge weights changed from minimum to maximum, which cause the inaccuracy of the modified correlation MST.

## 8. Generative Model for vine cluster graph

The nodes in the graph have three states: backbone, branch and leaf.

1. First we start with two backbone nodes and an edge between them. The two nodes grow towards two different directions.
2. **Starting from a backbone node to grow:** i) It can grow another backbone node with probability 0.58. In the case, the old backbone node stops to grow (deactivated), and the new backbone node is activated to grow. ii) It can grow a branch node with probability 0.4. In this situation, the old backbone node is still activated, which is allowed to further grow. The new branch node is also activated to grow. iii) It can grow a leaf node with probability 0.02. Then this backbone node gets deactivated and the new leaf node is not allowed to grow.
3. **Starting from a branch node to grow:** i) It can grow another branch node with probability 0.6. Then the old branch node gets deactivated and the new one get activated. ii) It can grow another leaf node. Then this branch node gets deactivated and the new leaf node is not allowed to grow.
4. **Leaf node is not allowed to grow.**
5. Keep on growing until no backbone node and no branch node is activated.

Here in our plot. Black nodes indicate backbone nodes. Brown stands for branch nodes and Green stands for leaf nodes.

