Problem 1

Use the stock returns in DailyReturn.csv for this problem. DailyReturn.csv contains returns for 100 large US stocks and as well as the ETF, SPY which tracks the S&P500.

Create a routine for calculating an exponentially weighted covariance matrix. If you have a package that calculates it for you, verify that it calculates the values you expect. This means you still have to implement it.

Vary λ ∈ (0, 1). Use PCA and plot the cumulative variance explained by each eigenvalue for each λ chosen.

What does this tell us about values of λ and the effect it has on the covariance matrix?

Analysis

Firstly, I tried to generate the exponentially weighted covariance matrix. Since the data provided only contains 60 days' information, for limited time period, I assigned the normalized weights which sum to 1 to the returns from 60 trading days. The weights are:

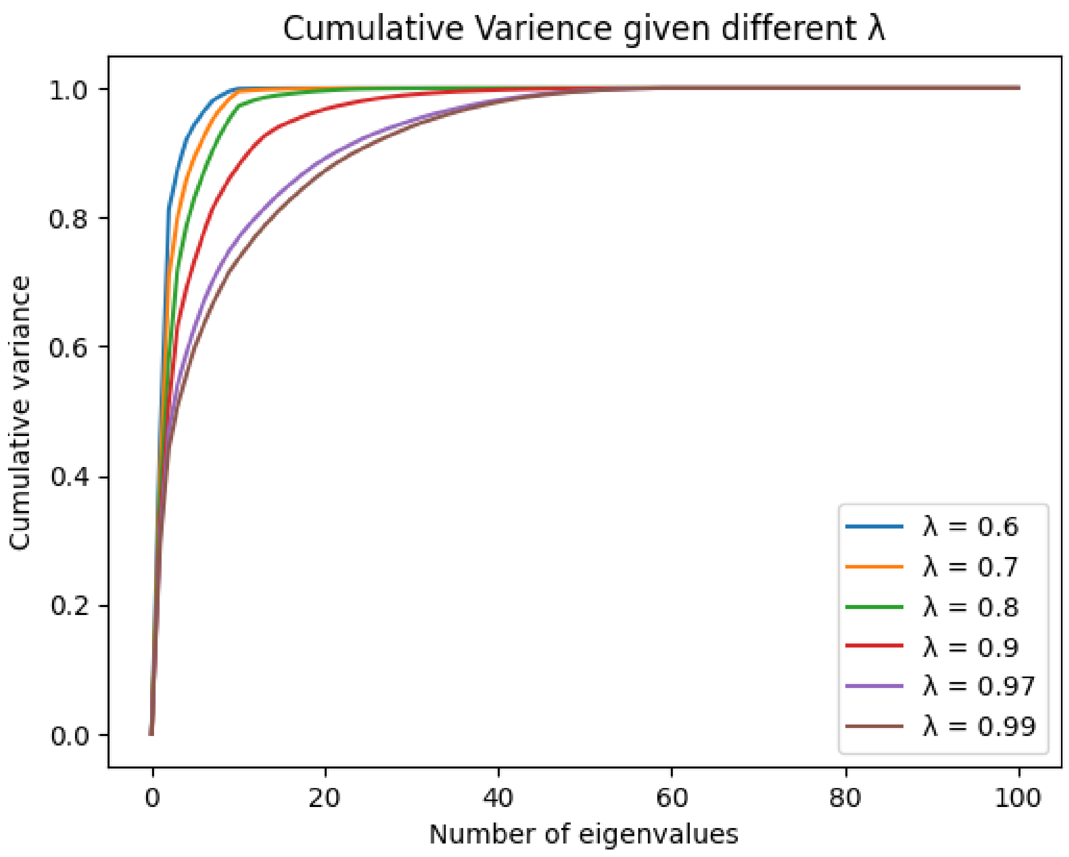
$$\widehat{W_{t-\iota}} = \frac{w_{t-i}}{\sum_{j=1}^{n} w_{t-j}}$$

So, the exponentially weighted covariances between any two of the stocks are:

$$\widehat{cov(x,y)} = \sum_{i=1}^{n} w_{t-i}\,(x_{t-i} - \bar{x})(y_{t-i} - \hat{y})$$

Since I have limit time period of data, I didn't use iteration to calculate the covariances.

Then, I performed PCA to get cumulative variance explained by each eigenvalue given each λ, and plotted the graph with number of counted eigenvalue in x axis and cumulative variance explained in y axis. Every line represents the condition of λ chosen.



It can be seen from the graph that given different λ, they all start at the same point and, as the number of counted eigenvalue going larger, converge to one point. Given the same number of counted eigenvalue, those covariance matrix with larger λ has low cumulative variance explained and the line of those covariance matrix with larger λ changes more gradually than those covariance matrix with smaller λ. The reasons are as follow. A smaller λ means the most recent data can have more weight and the weight decays more sharply as time going to earlier. This means latest data can explain a lot while early data accounts little. A larger λ means the most recent data can have relatively average weight as the data earlier. The weight decays more gradually. This means the difference between the variance explained by latest data and the variance explained by early data is small. So, the early data accounts relatively more.

Problem 2

Copy the chol_psd(), and near_psd() functions from the course repository – implement in your programming language of choice. These are core functions you will need throughout the remainder of the class.

Implement Higham’s 2002 nearest psd correlation function.

Generate a non-psd correlation matrix that is 500x500.

Use near_psd() and Higham's method to fix the matrix. Confirm the matrix is now PSD.

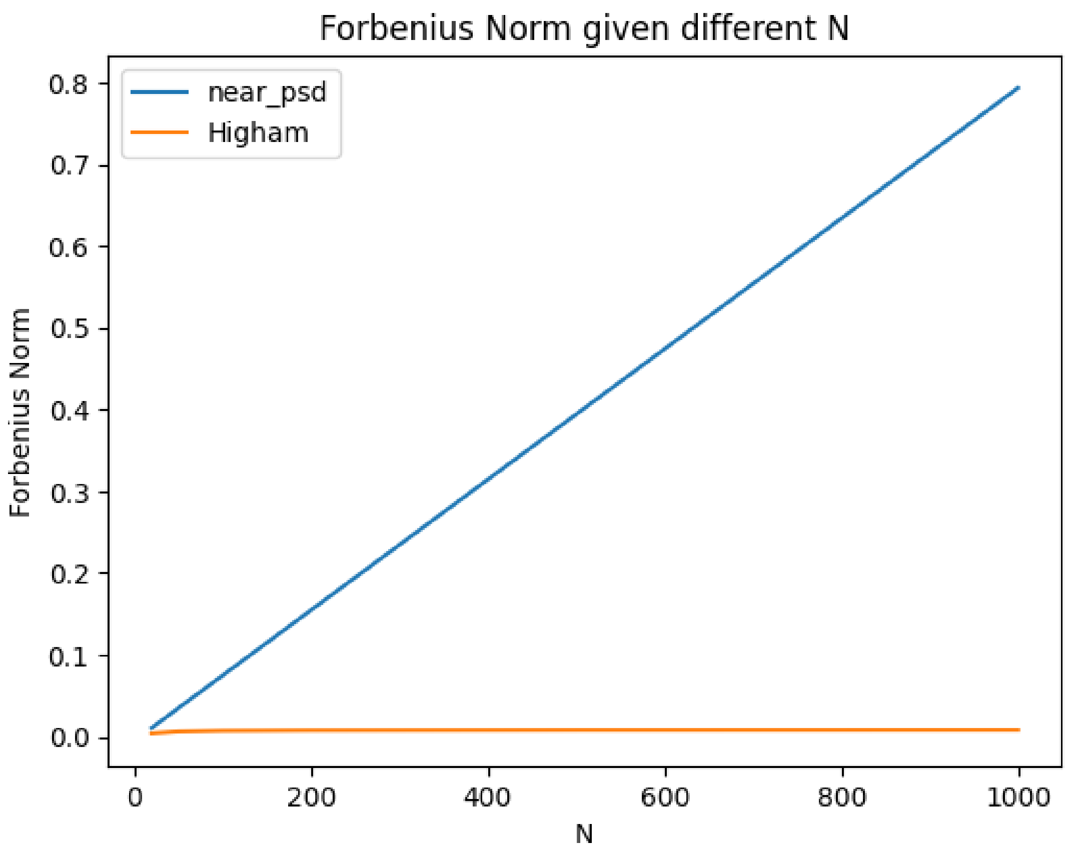
Compare the results of both using the Frobenius Norm. Compare the run time between the two. How does the run time of each function compare as N increases?

Based on the above, discuss the pros and cons of each method and when you would use each. There is no wrong answer here, I want you to think through this and tell me what you think.

Analysis

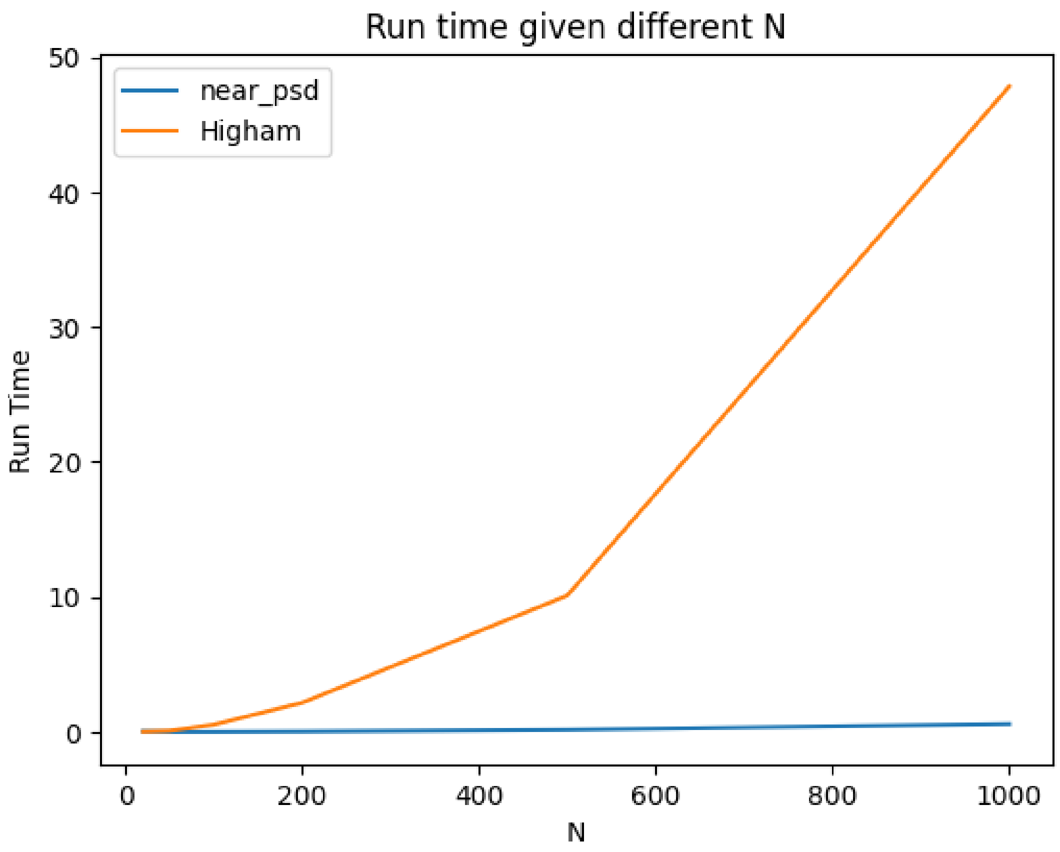
Firstly, I translated chol_psd(), and near_psd() Julia functions into Python functions. And wrote a Higham's 2002 nearest psd correlation function in Python. I generated a non-psd correlation metrix using the code given. After using near_psd() and Higham's method to fix the non-psd matrix, I have confirmed that the non-psd matrix has become PSD.

Then, by scaling the matrix size, I compared the Frobenius Norm between using near_psd() and Higham's method to fix the non-psd matrix. I made plot to show the comparison more intuitively.



It can be seen from the graph that as the size of matrix going large, the Frobenius Norm between the fixed matrix and the original matrix goes large linearly when the non-psd matrix is fixed by near_psd(), while the Frobenius Norm stays at a same low level when the non-psd matrix is fixed by Higham's method.

Then, by scaling the matrix size, I compared the run time between using near_psd() and Higham's method to fix the non-psd matrix. I made plot to show the comparison more intuitively.



It can be seen from the graph that as the size of matrix going large, the run time of fixing non-psd matrix by Higham's method goes longer sharply, while the run time of fixing non-psd matrix by near_psd() stays at a same low level.

The advantage of using near_psd() to fix non-psd matrix is that it can save time, while disadvantage is that it can lead to relatively large error. The advantage of using Higham's method to fix non-psd matrix is that it can lead to relatively small error, while disadvantage is that it costs longer time. For me, when the size of non-psd matrix is small, I prefer to use Higham's method because the difference of the run time is small and the result fixed by Higham's method is more accurate. However, when the size of non-psd matrix is large, I will decide based on whether the accuracy of the matrix matters. If I only want a psd matrix and it is doesn't matter whether this matrix must be like the original one, I will use near_psd() to save time. If it demands high likelihood of the matrix, I will use Higham's method.

Problem 3

Using DailyReturn.csv.

Implement a multivariate normal simulation that allows for simulation directly from a covariance matrix or using PCA with an optional parameter for % variance explained. If you have a library that can do these, you still need to implement it yourself for this homework and prove that it functions as expected.

Generate a correlation matrix and variance vector 2 ways:
1. Standard Pearson correlation/variance (you do not need to reimplement the cor() and var() functions).
2. Exponentially weighted $\lambda = 0.97$

Combine these to form 4 different covariance matrices. (Pearson correlation + var()), Pearson correlation + EW variance, etc.)

Simulate 25,000 draws from each covariance matrix using:
1. Direct Simulation
2. PCA with 100% explained.
3. PCA with 75% explained.
4. PCA with 50% explained.

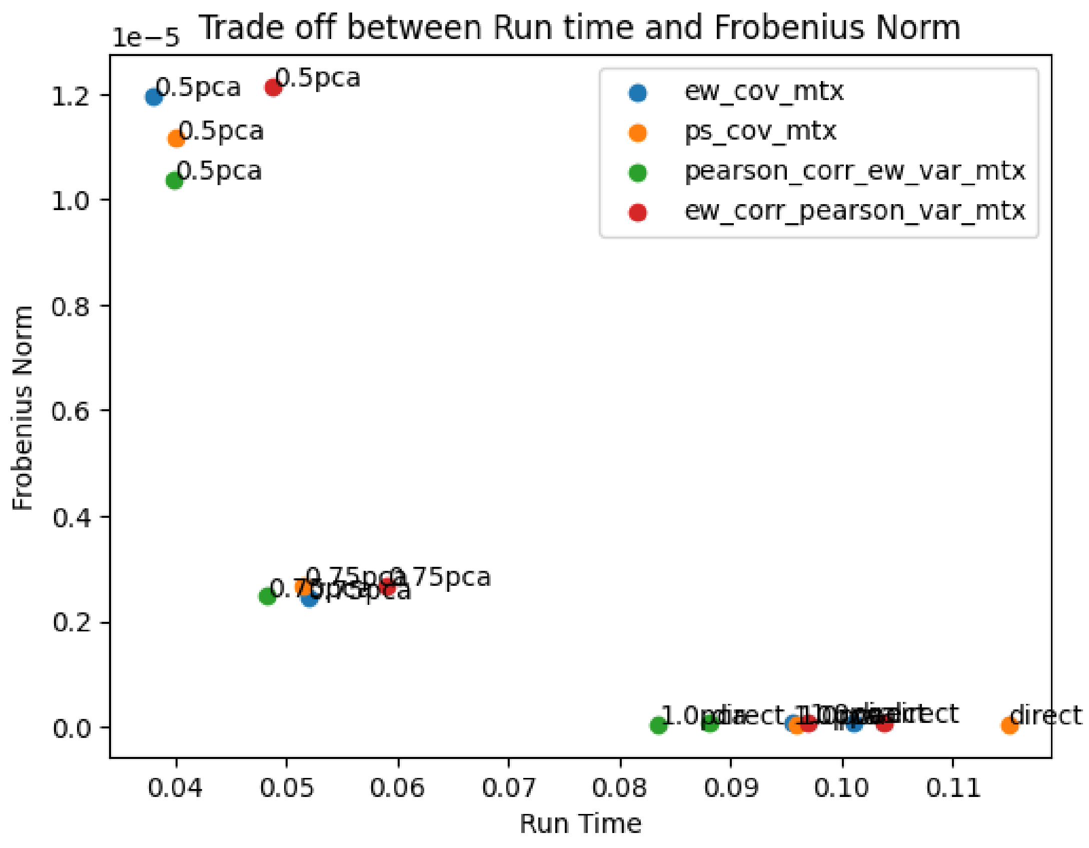
Calculate the covariance of the simulated values. Compare the simulated covariance to its input matrix using the Frobenius Norm (L2 norm, sum of the square of the difference between the matrices). Compare the run times for each simulation.

What can we say about the trade-offs between time to run and accuracy

Analysis

Firstly, I implement a multivariate normal simulation using Cholesky Factorization and using PCA with an optional parameter for % variance explained. I generated 4 covariance matrixes using two ways, Standard Pearson and Exponentially weighted. The 4 matrixes are respectively:
ew_cov_mtx,
ps_cov_mtx,
$pearson\_corr\_ew\_var\_mtx$,
$ew\_corr\_pearson\_var\_mtx$

Then I simulate 25,000 draws from each covariance matrix using Direct Simulation, PCA with 100% explained, PCA with 75% explained and PCA with 50% explained. I calculated their covariance of the simulated values, compare the simulated covariance to its input matrix using the Frobenius Norm and the run times for each simulation. I plotted the results in one scatterplot with run time on x axis and Frobenius Norm



It can be seen from the graph that generally, PCA simulation can shorten the run time by reducing working size of the matrix. But PCA simulation can sacrifice precision to trade back the short run time. For PCA, the lower percent of variance explained is, the shorter time the simulation lasts and the larger error the simulation has. On the other hand, the higher percent of variance explained is, the longer time the simulation lasts and the smaller error the simulation has. Direct simulation lasts longest and has the same accuracy as the PCA simulation with variance explained being100%, but it needs a little more time to simulate, which may be caused by the complexity of Cholesky factorization.

Speed and accuracy cannot be combined. Speed comes at the expense of precision and accuracy comes at the expense of speed. But it can be seen that PCA simulation with variance explained being in high range can save time without losing too much accuracy compared with direct simulation.