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?

Answer:

For how to generate the exponentially weighted covariance matrix, we talk about two ways to implement it in the lecture. The first one is implemented directly with the simple exponential smoothing model, where the covariance matrix is updated cyclically at all time steps and the exponential weighting factors are applied to the previous estimates of the covariance matrix at each iteration.

$$\sigma_t^2 = \lambda\sigma_{t-1}^2 + (1-\lambda)(x_{t-1} - \bar{x})^2$$

This approach is relatively more suitable for real-time updating of the covariance matrix, i.e., the return data obtained covers a large and increasing time range. In this case, it is sufficient to keep the matrix updated each time. It is a relatively more efficient and accurate loop.

For a limited data set, it is more suitable to first calculate the weights for each

time step, using the following formula, and then normalize the results and finally calculate the new covariance matrix.

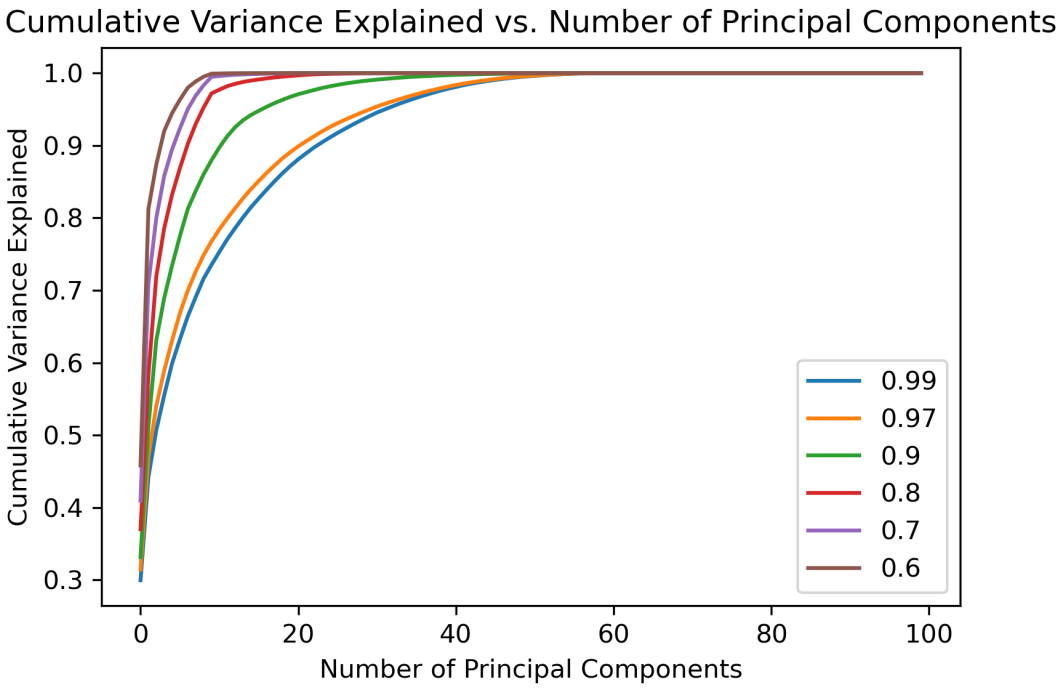
$$w_{t-i} = (1-\lambda)\lambda^{i-1}$$

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

Since it makes the calculated weights closer to the desired situation by normalization when faced with a limited number of data samples. Therefore, it may be more applicable in this question.

Of course, both methods give more consistent answers when faced with large data samples, as verified in a randomly generated matrix of 1000x100.

Next, PCA was used and plot the cumulative variance explained by each eigenvalue for each λ chosen. The results obtained are in the following figure. From the results, it can be seen that the curve is smoother when λ is larger, while the curve is smoother when λ is smaller.



This is because a larger λ means that the weights of the data converge to equal weights and more data are needed to explain the variance of the sample. while a smaller λ gives a higher weight to the data that are closer in time, which means that only a small amount of data is needed to explain the variance of the sample.

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.

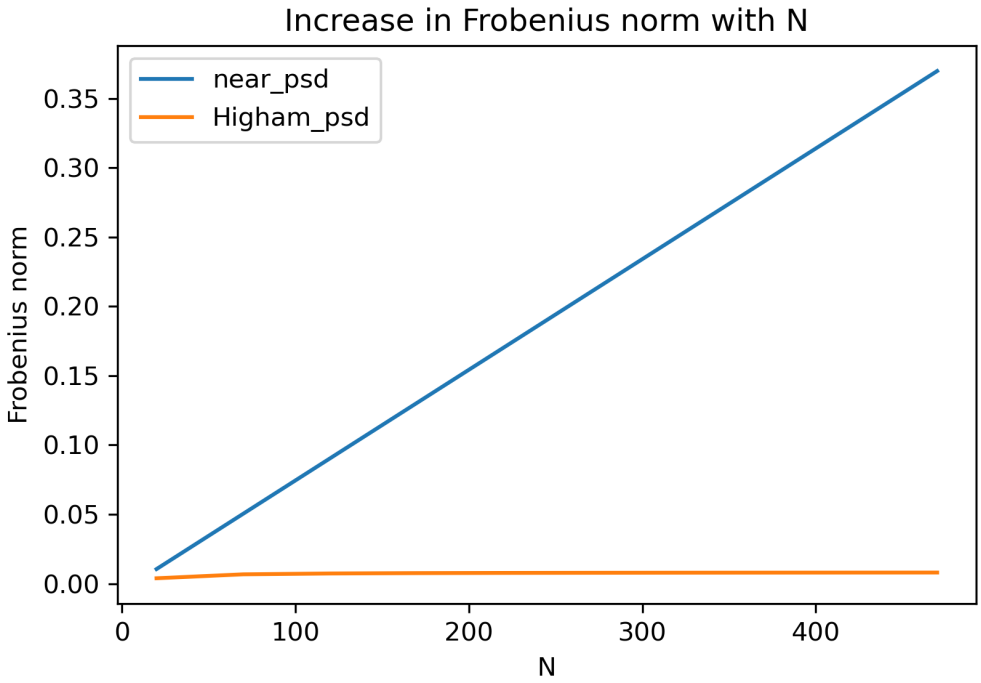
Use near_psd() and Higham's method to fix the matrix. Confirm the matrix is now PSD.

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?

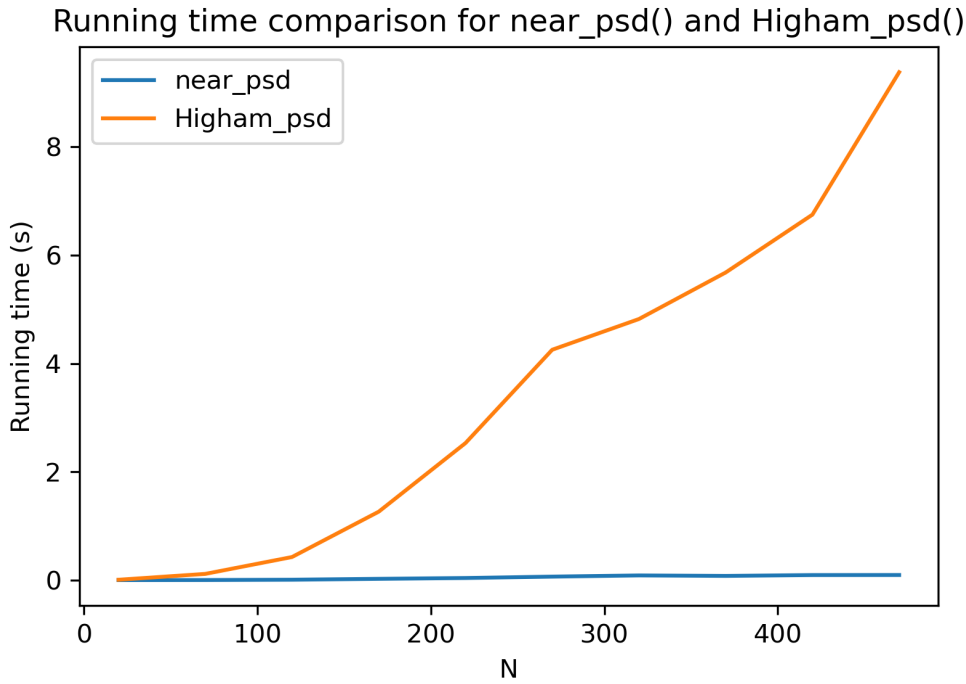
Based on the above, discuss the pros and cons of each method and when you would use each. There is no wrong answer here, I want you to think through this and tell me what you think.

Answer:

By applying two different methods of finding semi-positive definite matrices and calculating the Frobenius Norm obtained by both methods with increasing N. The results are in the following. As can be seen from the figure, the results given by near_psd(), Norm increases in a straight line. While the matrix computed by the Higham method, Norm can be maintained at a lower level all the time, which means that its result is closer to the original matrix.



And comparing the running time required by the two methods, the running time of near_psd() is always maintained at a low level. The runtime of the Higham method, on the other hand, grows approximately exponentially.



When the matrix is small, there is not much difference between near_psd()

algorithm and Higham algorithm for both runtime and accuracy. But when N is large enough, if my result is more about the speed of the operation, then I will choose near_psd(). And if I need the result to be accurate more, I will choose the Higham method. I might set a limit for myself though, when Norm is larger than a certain value, I might prefer to sacrifice the speed of the operation rather than the accuracy.

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.

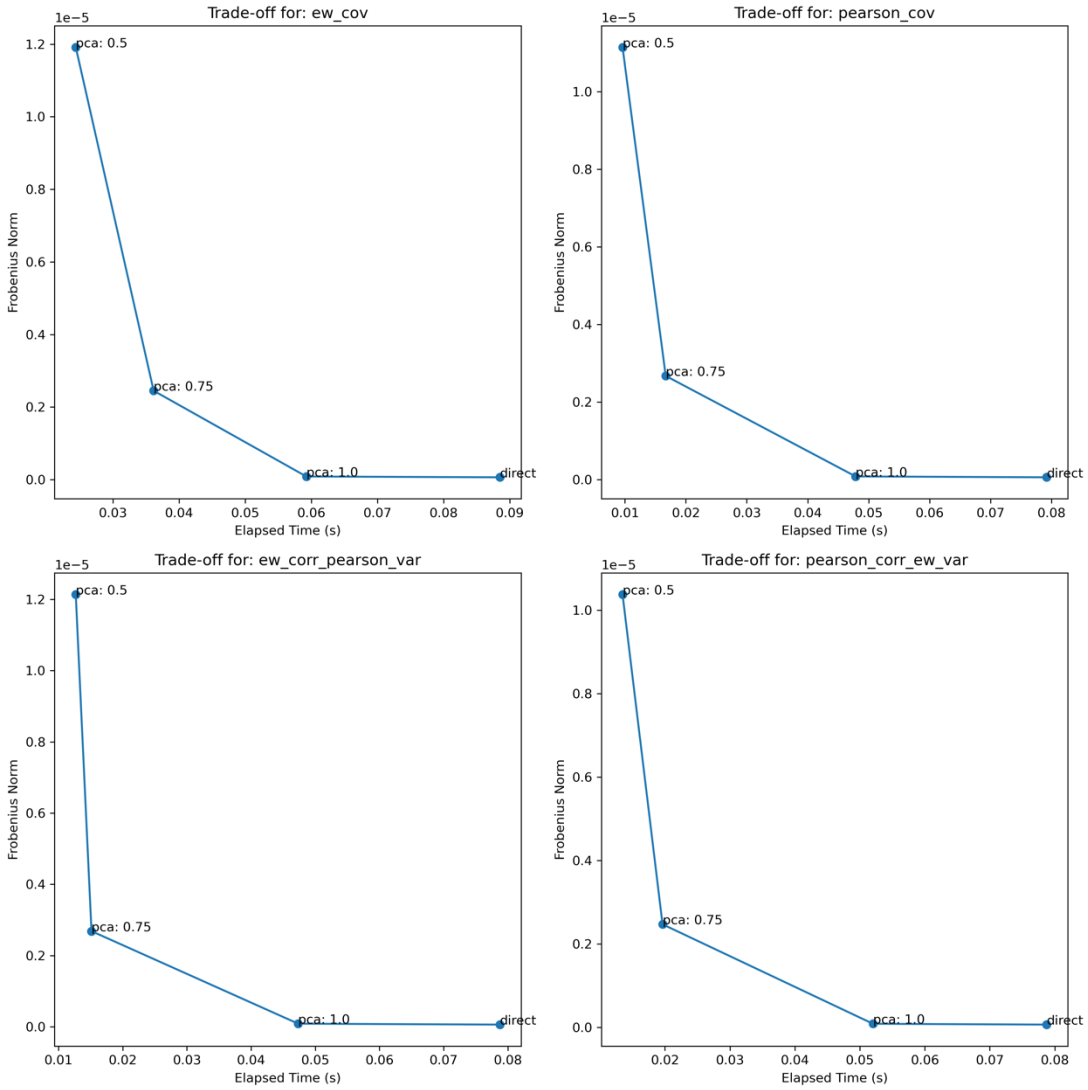
4. PCA with 50% explained.

Calculate the covariance of the simulated values. Compare the simulated covariance to it's 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.

Answer:

By combining the four covariance matrices generated, I applied multivariate normal simulation to produce the results, i.e., time and accuracy, in the figure below.



PCA speeds up operations by reducing dimensions, but also loses more information. This is shown by the fact that as the explained variance is reduced, the speedup is accompanied by an increase in Norm. On the other hand, for the comparison between PCA simulations with an explain variance of 1 and direct simulations, the PCA method gives faster results and the accuracy is not much different. This is because both retain essentially the same information,

while the Cholesky decomposition requires the use of loops to compute the matrix, which can slow down the speed.

Regarding the balance between the two, for me, I would probably prefer to use the PCA method with an explained variance of 0.75 to 1 for the simulation, which retains more of the needed information while taking into account the speed of the calculation. Still, the direct simulation method has more advantages in accuracy than the PCA method, and it has its own advantages as well when necessary.