Project Week 03

Peiyi Xu

Problem 1

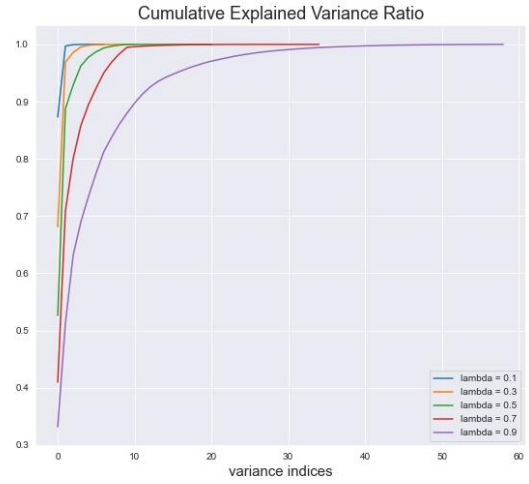
Use the stock returns in DailyReturn.csv for this problem. DailyReturn.csv contains returns for 100 large US stocks and as well as the ETF, SPY which tracks the S&P500.

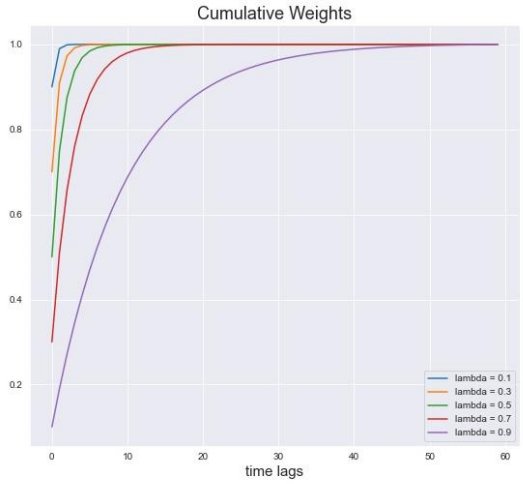
Create a routine for calculating an exponentially weighted covariance matrix. If you have a package that calculates it for you, verify that it calculates the values you expect. This means you still have to implement it.

Vary $\lambda \in (0, 1)$. 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 1

We can plot the cumulative ratio of variance explained by eigenvalues and the cumulative weights from $t-1$ to $t-60$ when $\lambda = 0.1, 0.3, 0.5, 0.7, 0.9$. We can see from the second graph that the less λ is, the greater weight we put on current information; The greater λ is, the weights are more equally distributed on information of each time periods. Therefore, when λ is closer to 0, we need less principal components to explain for the variance of the exponentially weighted covariance matrix we calculated; when λ is closer to 1, we will need 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.

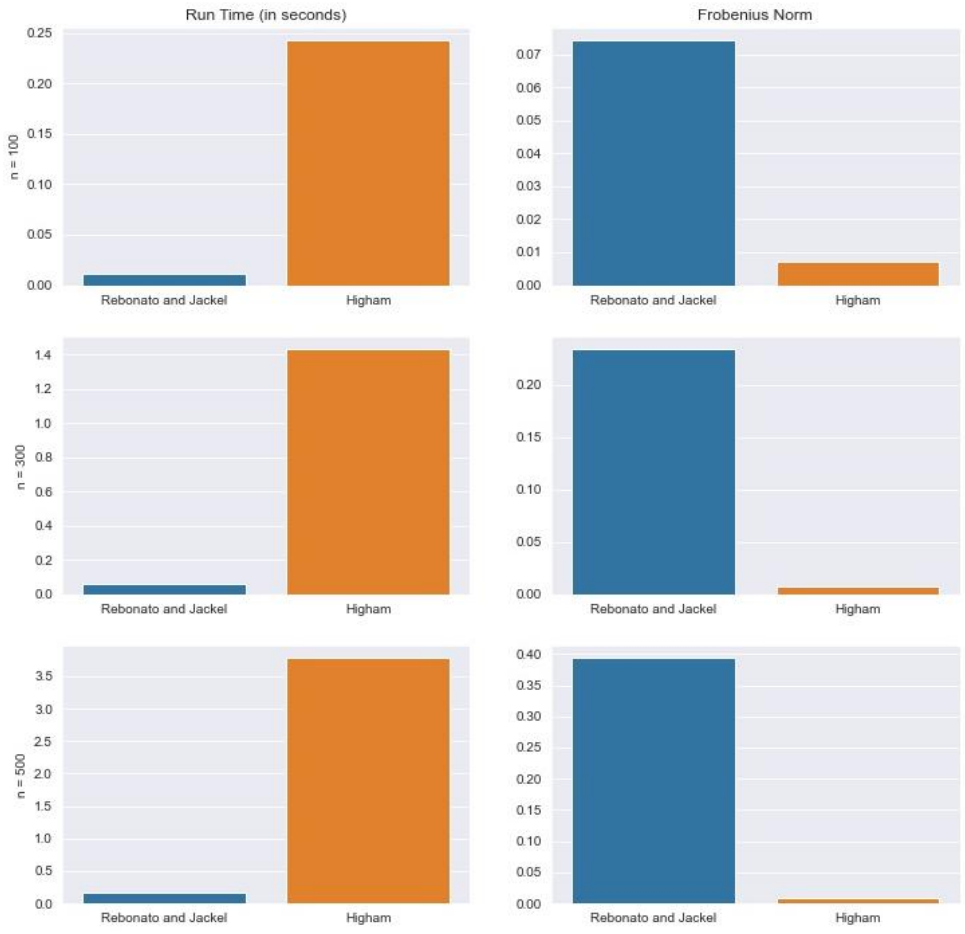
Generate a non-psd correlation matrix that is 500x500. Use near_psd() and Higham's method to fix the matrix. Confirm the matrix is now PSD.

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.

Answer 2

We can calculate the eigen values of the correlation matrix after fixing it with Rebonato & Jackels method and Higham's method. We confirmed that there are no negative eigen values and thus fixed matrices are PSD.

The following graph shows the run time and Frobenius Norm for $n = 100, 300, 500$. We can see that the run time of Rebonato & Jackel's algorithm are basically the the same, while those of Higham's increases at a nearly quadratic rate. On the other hand, the Frobenius Norm of Higham's algorithm changed very little, but that of Rebonato & Jackel's algorithm increased rapidly.



We may conclude that Rebonato and Jackel's method is a fast but inaccurate one, while Higham's method is an accurate but slow one. When dealing with matrices of small sizes there may not be too much difference, as both can be fast and accurate enough; but when it comes to deal with matrices with large sizes, one must choose between accuracy and efficiency.

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?

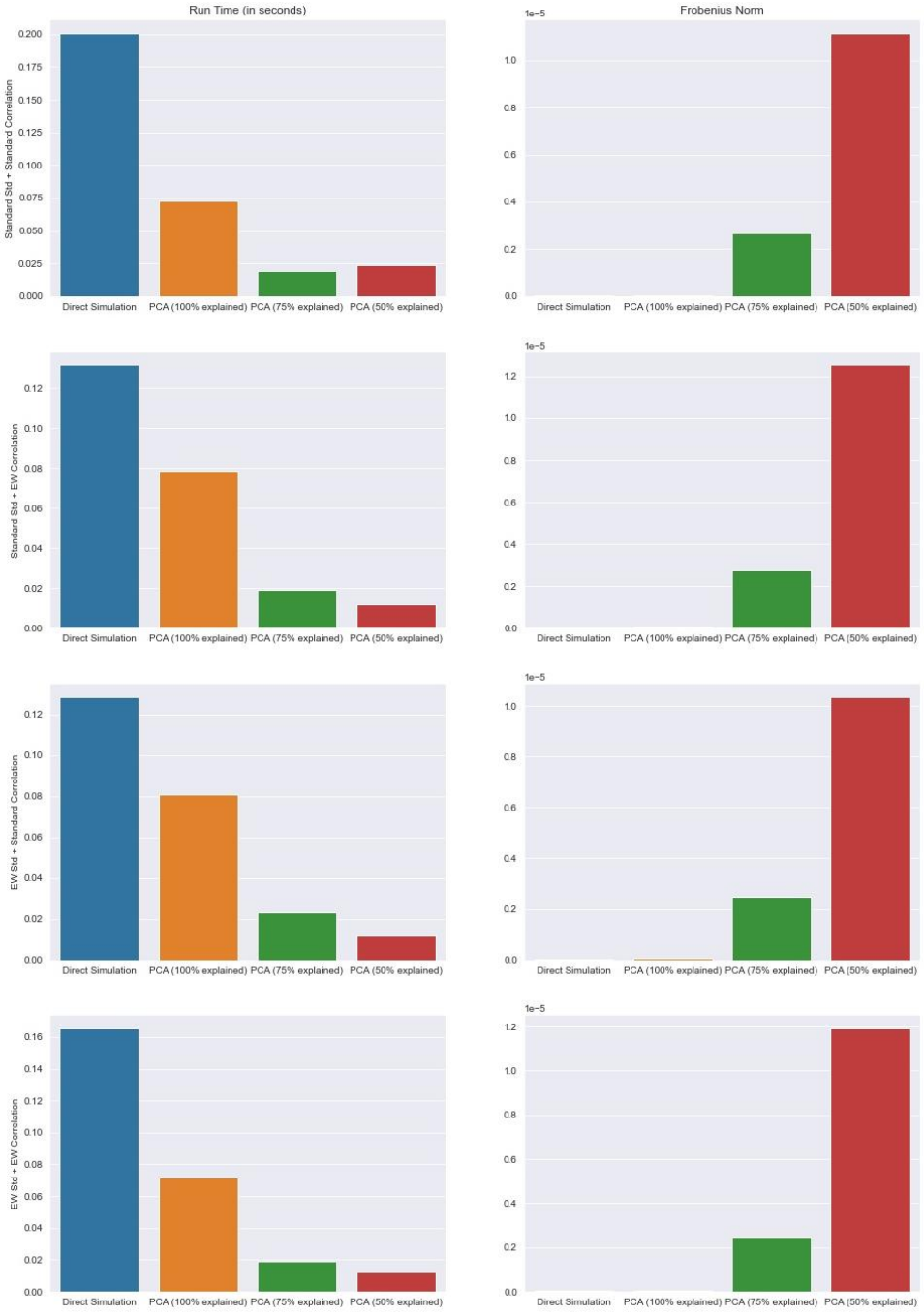
Answer 3

The following tables and graph shows the run time and Frobenius Norm of the four simulation methods applied on the covariance matrix generated by:

1. standard deviation & Pearson correlation
2. standard deviation & exponentially weighted Pearson correlation
3. exponentially weighted standard deviation & Pearson correlation
4. exponentially weighted standard deviation & exponentially weighted Pearson correlation

Run Time (seconds)	*Direct Simulation*	*PCA 100% explained*	*PCA 75% explained*	*PCA 50% explained*
std & correlation	0.126219	0.064827	0.018953	0.010971
std & EW correlation	0.126661	0.063829	0.017951	0.010972
EW std & correlation	0.129653	0.062834	0.017952	0.011969
EW std & EW correlation	0.149073	0.068400	0.017954	0.013962

Frobenius Norm	*Direct Simulation*	*PCA 100% explained*	*PCA 75% explained*	*PCA 50% explained*
std & correlation	4.866301	6.633571	270.392342	1114.777441
std & EW correlation	4.488397	4.954941	283.530388	1254.964411
EW std & correlation	7.057699	4.196981	249.222515	1042.556673
EW std & EW correlation	4.547950	5.540411	245.092077	1191.059750



We discover that though we can be 10 times faster on generating random simulations using PCA with 75% and 50% variance explained, we end up having hundreds times larger Frobenius Norms. We don't necessarily gain improvements in efficiency in proportion to the sacrafice we made on accuracy.