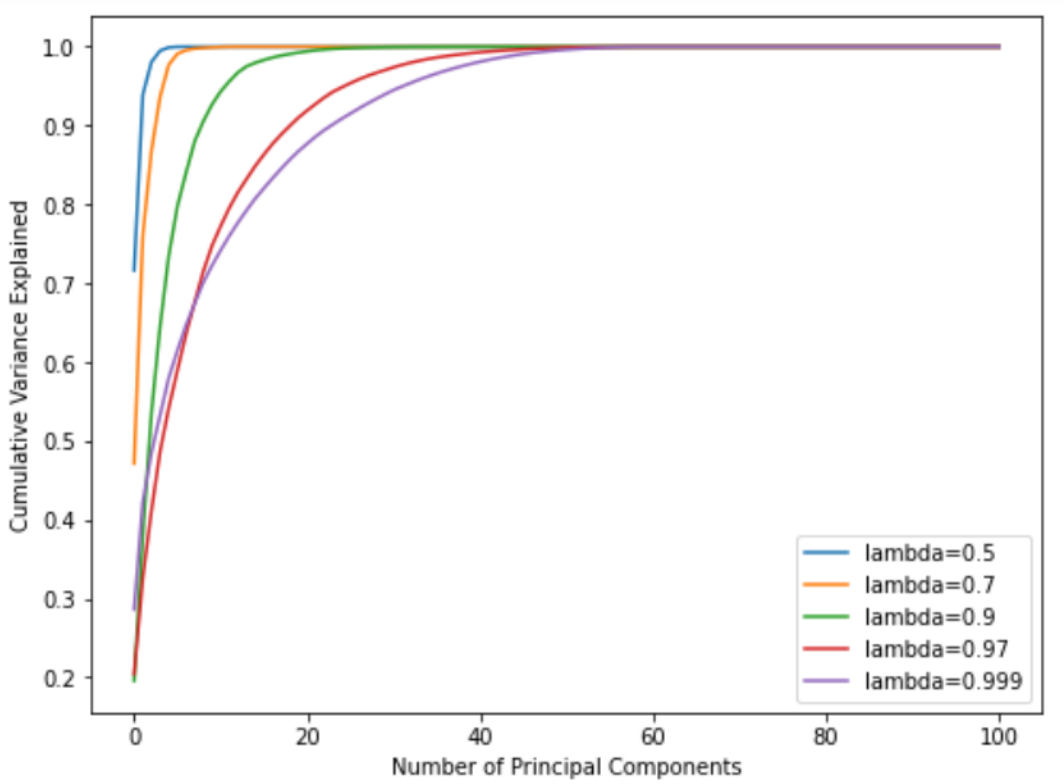
Problem 1:

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:



The plot shows the cumulative variance explained by each eigenvalue for different values of λ. It shows how much of the total variation in the data is captured by the first n principal components, where n is the number of eigenvalues.

The value of λ determines the amount of weight given to more recent returns in the calculation of the covariance matrix. If λ is close to 1, more weight is given to recent returns, which means that the covariance matrix is more sensitive to changes in the most recent returns. If λ is close to 0, more weight is given to older returns, and the covariance matrix is less sensitive to changes in the most recent returns.

From the plot, we can see that as the value of λ increases, the cumulative variance explained by the first few eigenvalues increases as well, which means that a smaller number of principal components is needed to capture most of the variation in the data. This suggests that when λ is close to 1, the covariance matrix is dominated by more recent returns, and the variation in the data can be captured by a small number of principal components. On the other hand, when λ is close to 0, a larger number of principal components is needed to

capture most of the variation in the data, which means that the covariance matrix is more influenced by older returns and that the variation in the data is more complex.

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. You can use the code I used in class:

```
n=500
sigma = fill(0.9,(n,n))
for i in 1:n
sigma[i,i]=1.0
end
sigma[1,2] = 0.7357
sigma[2,1] = 0.7357
```

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.

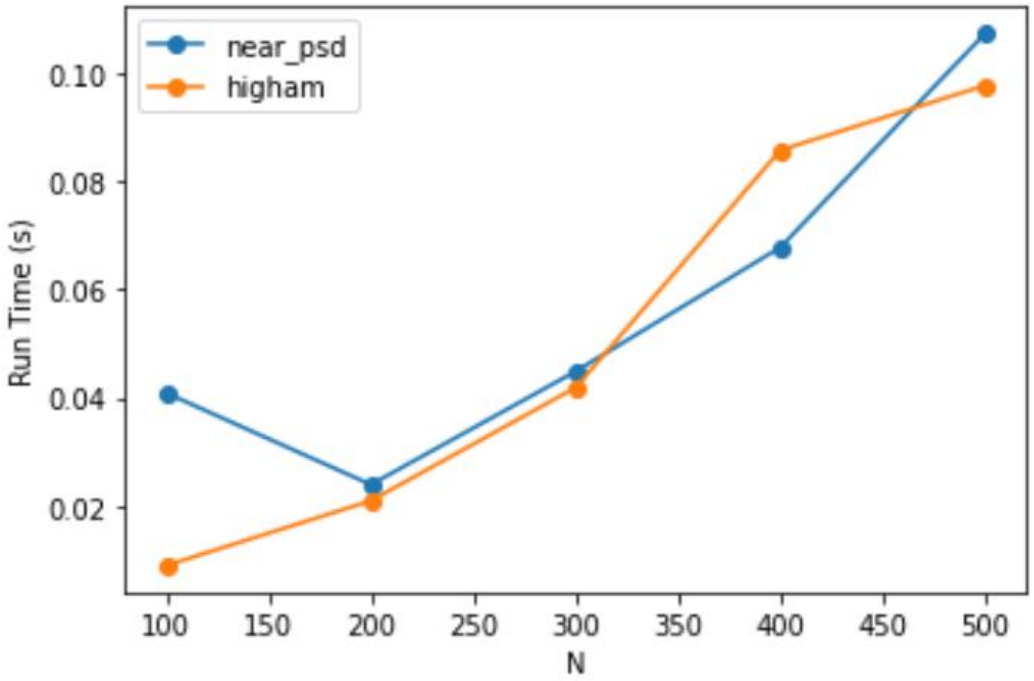
Answer：

The run time of near_psd() and higham() will increase as N increases because the algorithm needs to perform operations on a larger matrix. However, the exact time will depend on the specific implementation and the machine running the code. Their run times are both very short (at least for a 500*500 matrix) and there is very little difference between them. So whenever we run them, we may get plots that look quite different.

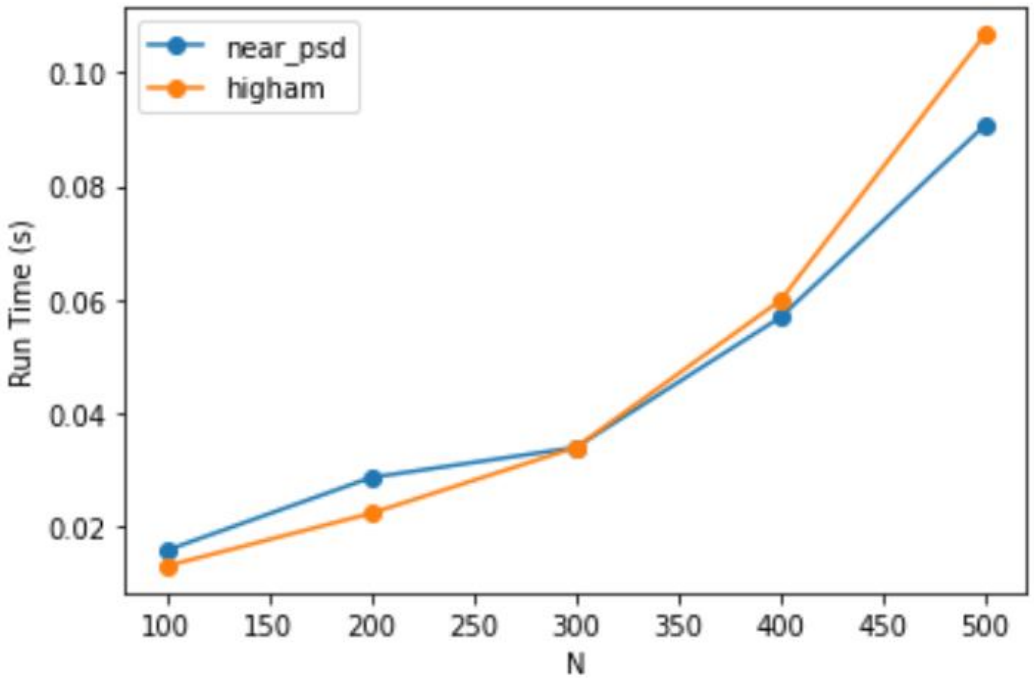
For near_psd(), the pros are that the implementation is relatively simple and it works well for small matrices. The cons are that it may not work well for large matrices because it only sets negative eigenvalues to 0 and not to a positive value. It may also lead to over-correction, making the matrix more positive definite than necessary.

For higham(), the pros are that the implementation is more robust and it works well for large matrices and those with many negative eigenvalues. The cons are that the implementation is more complex than near_psd(). It may also be slow for very large matrices.

```
Time taken by near_psd for N = 100 is: 0.04089045524597168 seconds
Time taken by higham for N = 100 is: 0.00897359848022461 seconds
Time taken by near_psd for N = 200 is: 0.023936986923217773 seconds
Time taken by higham for N = 200 is: 0.02095651626586914 seconds
Time taken by near_psd for N = 300 is: 0.04488015174865723 seconds
Time taken by higham for N = 300 is: 0.041887521743774414 seconds
Time taken by near_psd for N = 400 is: 0.06781840324401855 seconds
Time taken by higham for N = 400 is: 0.08577132225036621 seconds
Time taken by near_psd for N = 500 is: 0.10734272003173828 seconds
Time taken by higham for N = 500 is: 0.09773850440979004 seconds
```




```
Time taken by near_psd for N = 100 is: 0.0157487392425537 seconds
Time taken by higham for N = 100 is: 0.01300191879272461 seconds
Time taken by near_psd for N = 200 is: 0.02860879898071289 seconds
Time taken by higham for N = 200 is: 0.022289752960205078 seconds
Time taken by near_psd for N = 300 is: 0.03390908241271973 seconds
Time taken by higham for N = 300 is: 0.033896684646606445 seconds
Time taken by near_psd for N = 400 is: 0.056845903396606445 seconds
Time taken by higham for N = 400 is: 0.05982851982116699 seconds
Time taken by near_psd for N = 500 is: 0.09075784683227539 seconds
Time taken by higham for N = 500 is: 0.10671305656433105 seconds
```



I would choose to use near_psd() when I need to make a quick correction to a non-PSD matrix. It is not guaranteed to produce the closest PSD matrix, but it is guaranteed to produce a PSD matrix. I would choose to use higham() when I need to find the closest PSD matrix to the non-PSD matrix. It is guaranteed to produce the closest PSD matrix to the original non-PSD matrix, but it is more time-consuming to implement.

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:

The trade-off is that as we decrease the explained variance, we will save some time, but we will get less accurate results.
With Direct Simulation, we can get the most accurate results, as we are using the covariance matrix directly to generate the simulated values. But this approach may take a long time to run, especially if we have a large number of stocks and a long time period.
On the other hand, using PCA to simulate values reduces the computational time, as we are only using the principal components which capture the majority of the variation in the data.

As the explained variance decreases, the time taken to run the simulation decreases, but the difference between the simulated covariance and the input covariance matrix increases.
When the explained variance is 100%, the simulated covariance is very close to the input covariance matrix and the difference between the two matrices is small. This indicates that the PCA simulation with 100% explained variance accurately captures the relationships between the stocks.
As the explained variance decreases to 75% and 50%, the difference between the simulated covariance and the input covariance matrix increases, indicating that the relationships between the stocks are not captured as accurately.
Even though the PCA simulations with 75% and 50% explained variance are less accurate, they are still able to capture some of the relationships between the stocks and may still be useful for certain applications.