Week 3 assignment

---- Dong Pu (dp289)

Question1

- Question: 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 λchosen.
- What does this tell us about values of λ and the effect it has on the covariance matrix?

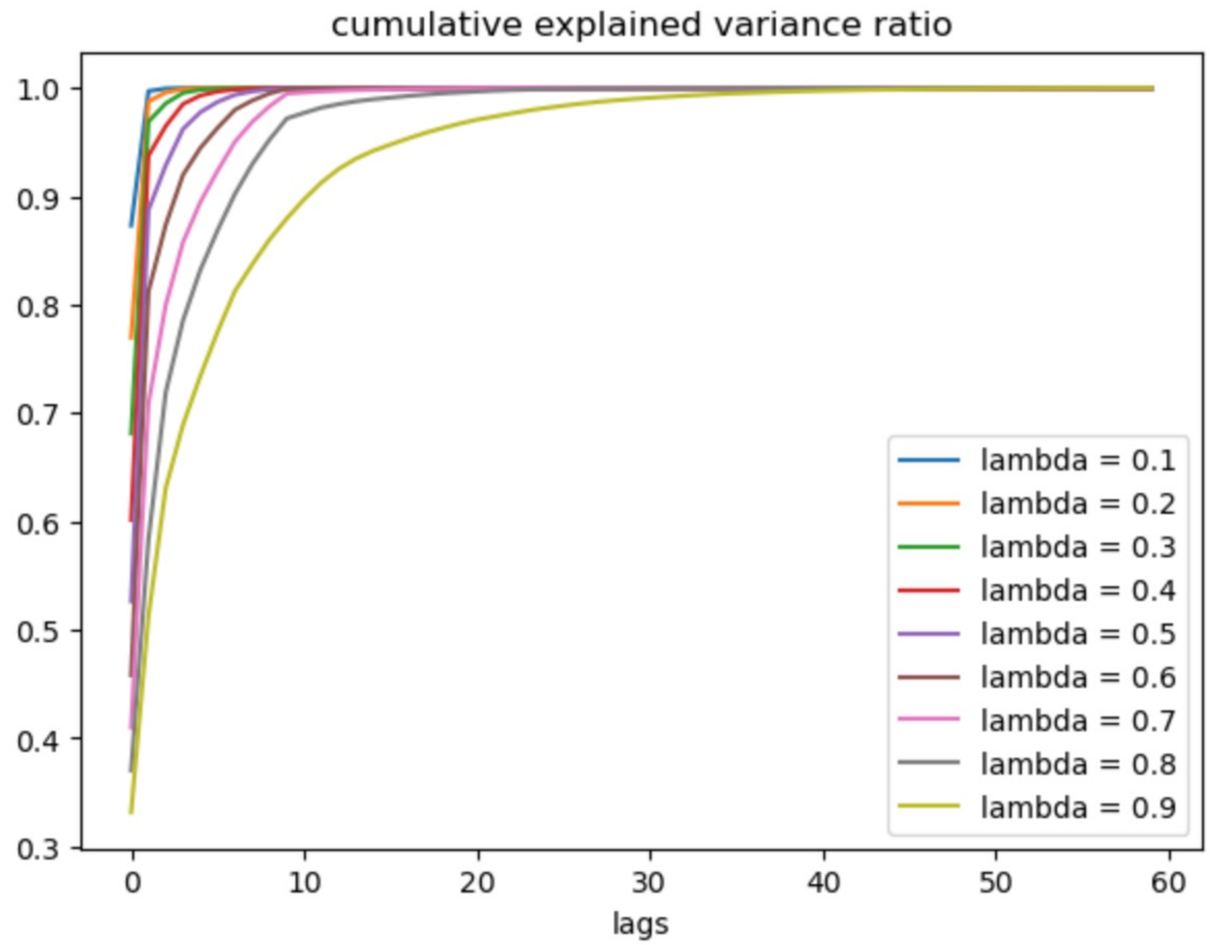
Question 1

- Answer:
- For exponentially weighted covariance matrix, we calculate the weights using the following formula. Then multiply the error matrix with the weights to get the covariance matrix.

$$w_{t-i} = (1 - \lambda)\lambda^{i-1} \qquad \widehat{w_{t-i}} = \frac{w_{t-i}}{\sum_{j=1}^{n} w_{t-j}}$$

Question 1

- Answer:
- For calculate the PCA cumulative explained variance ratio:
- When the lambda is small, the exponantially weighted covariance matrix puts more weights on the data, and decays quickly as lags increases. When lambda becomes bigger, the covariance matrix is more evenly distributed throughout the time, and decays slowly towards the max lag.

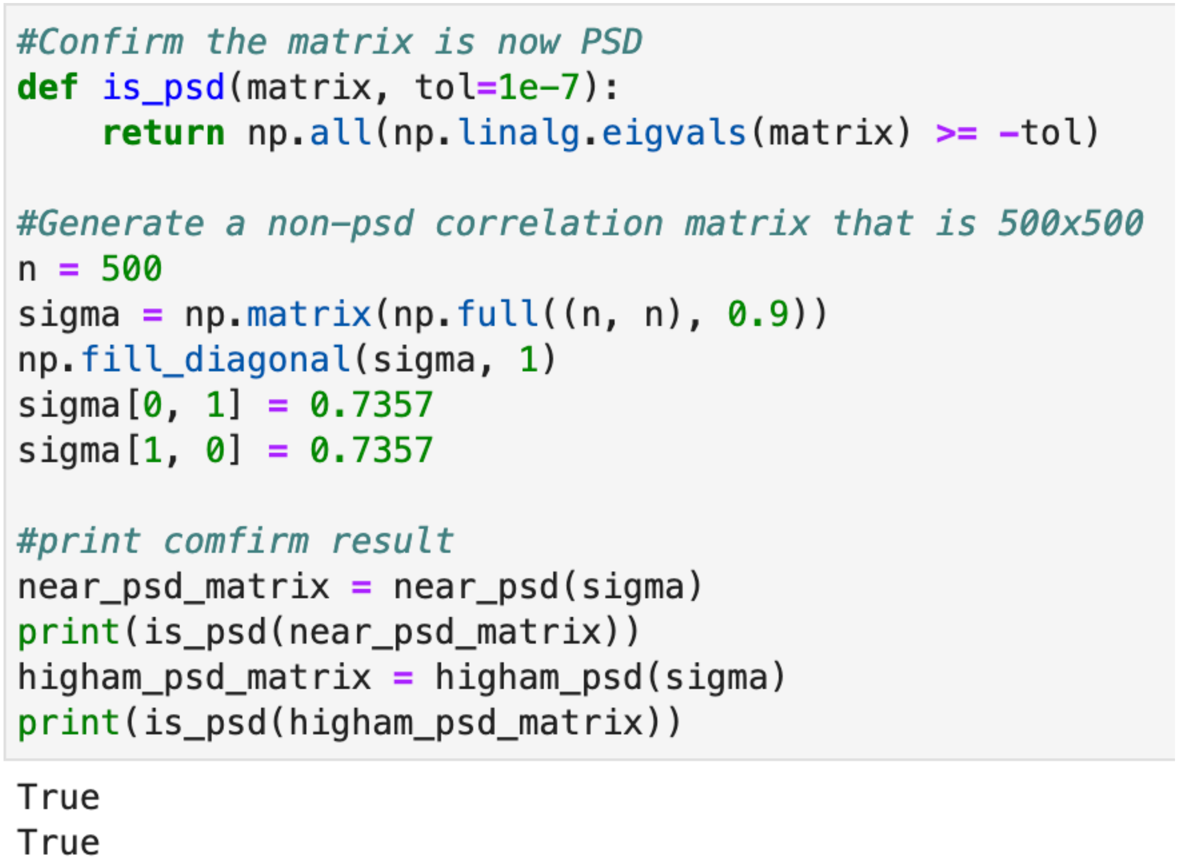


Question 2

- Question:
- 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.

Question 2

- Answer:
- From the code result, we can see that both matrix are now psd matrix.

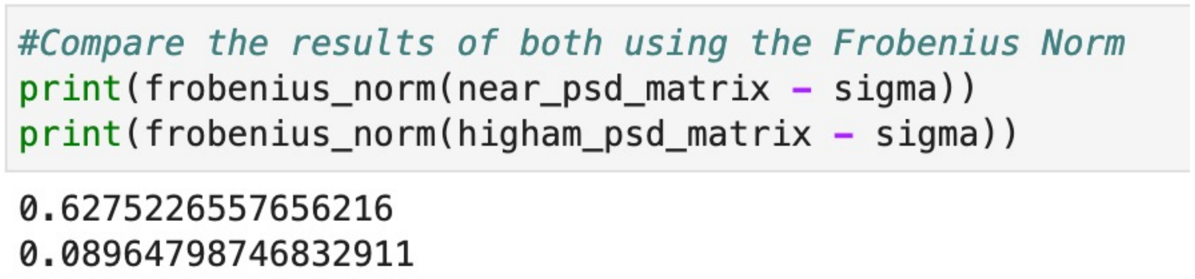


```
#Confirm the matrix is now PSD
def is_psd(matrix, tol=1e-7):
    return np.all(np.linalg.eigvals(matrix) >= -tol)

#Generate a non-psd correlation matrix that is 500x500
n = 500
sigma = np.matrix(np.full((n, n), 0.9))
np.fill_diagonal(sigma, 1)
sigma[0, 1] = 0.7357
sigma[1, 0] = 0.7357

#print comfirm result
near_psd_matrix = near_psd(sigma)
print(is_psd(near_psd_matrix))
higham_psd_matrix = higham_psd(sigma)
print(is_psd(higham_psd_matrix))
```

```
True
True
```

Question 2

- Answer:



```python
#Compare the results of both using the Frobenius Norm
print(frobenius_norm(near_psd_matrix - sigma))
print(frobenius_norm(higham_psd_matrix - sigma))
```

```
0.6275226557656216
0.08964798746832911
```

- Compare the Frobenius Norm, it shows that the result from higham_psd function is more accurate than the one from near_psd.

Question 2

- Answer:

Runtime	Near_psd	Higham_psd	ratio
100	0.00277590	0.02106475	7.588
500	0.05108690	0.41628170	8.148
900	0.14833307	1.72440409	11.625

- From the table above, we can see that higham_psd takes much longer time to compute than near_psd. As the runtime increases, the ratio bewteen these two functions increases.
- We can see that the near_psd function is less accurate than higham_psd, but it runs very fast. The higham_psd function is more accurate, but runs much slower.
- If we don’t have constraint on the runtime, we can pick higham_psd. If we have constraint on runtime and requirement for accuracy is not so strict, we can pick near_psd.

Question 3

- Question :
- Implement a multivariate normal simulation that allows for simulation directly from a covariance matrix or using PCA with an optional parameter for % variance explained.
- 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 $\lambda = 0.97$

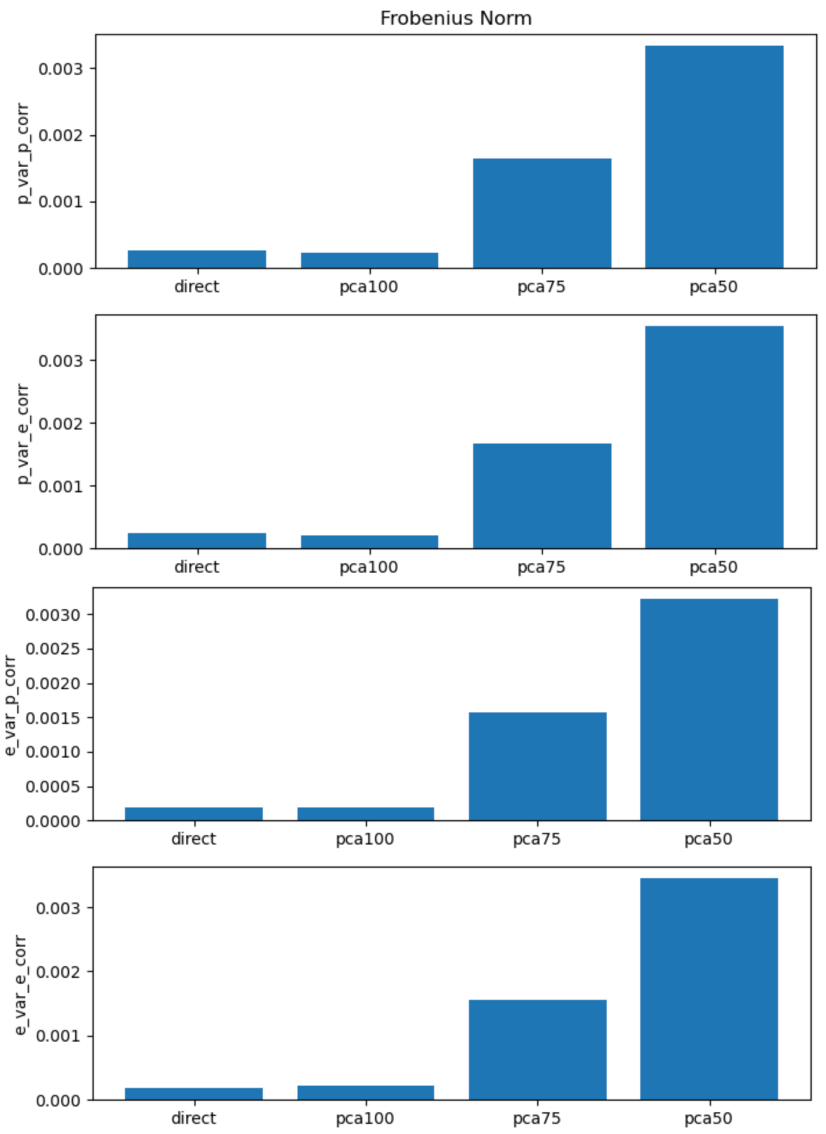
- Combine these to form 4 different covariance matrices. (Pearson correlation + var()), Pearson correlation + EW variance, etc.)
- Simulate 25,000 draws from each covariance matrix using:

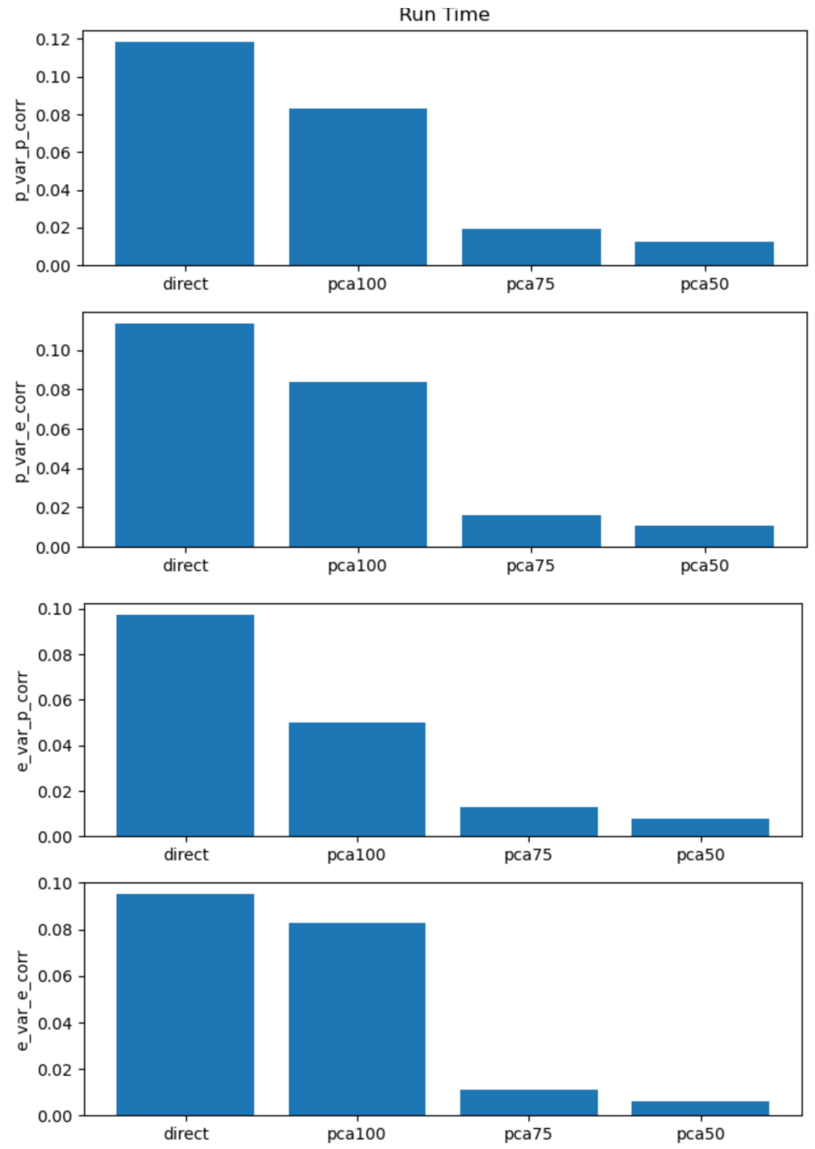
1. Direct Simulation
2. PCA with 100% explained.
3. PCA with 75% explained.
4. PCA with 50% explained.

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.

Question 3

- Answer:





Question 3

- Answer:
- From the graph shown above, as explained percentage decreases on PCA, the accuracy becomes worse while the runtime becomes faster. The advantage that the faster runtime brings cannot compensate the loss of the accuracy, which means the overall gain from decreasing PCA explained percentage is not good. I would suggest only do it when the computational resources is constrainted.