

Week 3 assignment

Jingyu Tan

JT393



Problem 1

- **Question:**
- 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?**

Problem 1

- In this problem, first we need to build up some functions to calculate weighted covariance matrix. Below, are the based formulas.

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

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

$$\widehat{\sigma}_t^2 = \sum_{i=1}^n w_{t-i} (x_{t-i} - \bar{x})^2$$

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



Problem 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.

Problem 2

- **Ans: First, we checked two functions returned matrix are now psd matrix.**

```
• #check if 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 confirm 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

Problem 2

- This shows that the result from `higham_psd_matrix` function is more accurate than the one from `near_psd_matrix` function.

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

✓ 0.0s

0.6275226557650357

0.08964799629214762

Problem 2

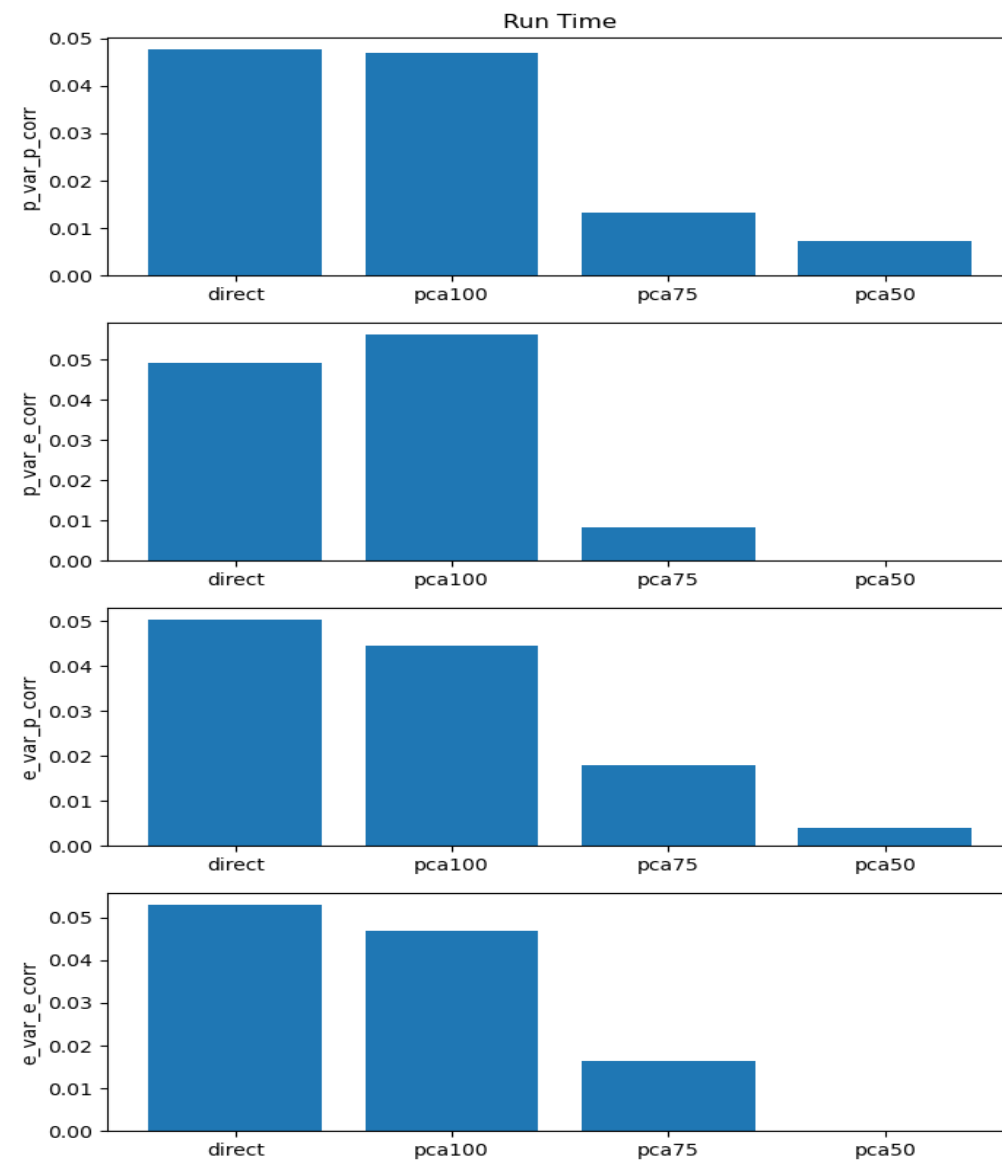
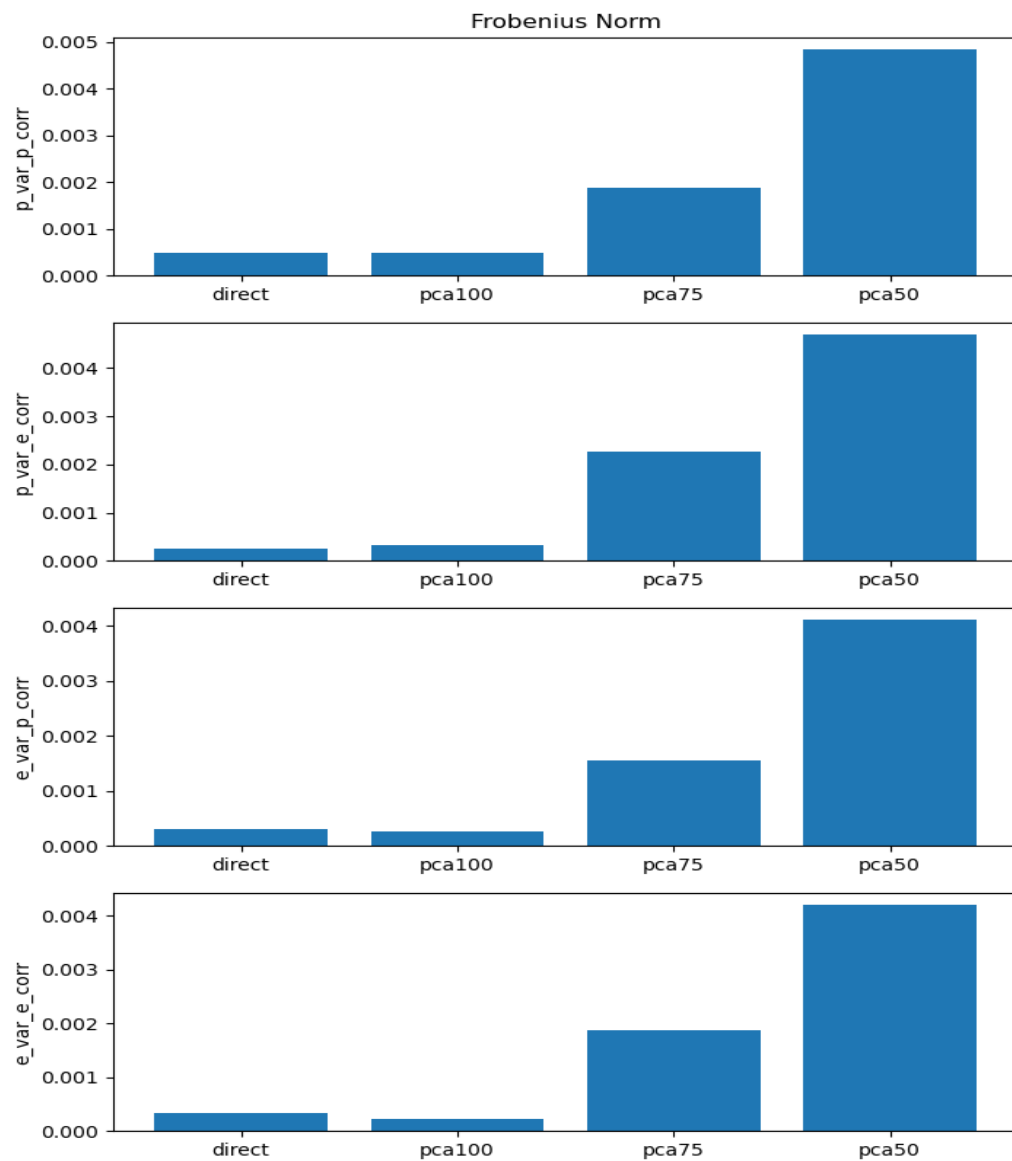
- **Ans:**
- Based on the graph, we can see that near_psd method requires much less time to run than higham_psd method. As the run time increase, the gap between these two methods also increase.
- We know that the near_psd function is less accurate than higham_psd, but it runs very fast. And higham_psd runs slower but more accurate.
- If we have to run it in a large number of times and we do have a time constraint, we should use near_psd. Otherwise, we should choose higham, which is because it is more accurate.

RUN TIME	Near_psd	Higham_psd	Ratio
300	0.0448442	0.807297945	18.0023
500	0.0623307	1.580256939	25.35278
700	0.0959609	3.069633007	31.98839

Problem 3

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

Problem 3



Problem 3

- **Ans:** Based on these graphs, we can see that as explained percentage decreases on PCA, the accuracy decreases while the runtime also decreases.
- I think the trade off is not fair. The advantage that the faster runtime brings cannot compensate the loss of the accuracy. This means that the overall gain from decreasing PCA explained percentage is not good.
- However, if we do have a time constraint that forces us cannot use more accurate method, we should choose to use PCA with less explained percentage.