Quantitative Risk Management Project 3

Qijun Yang
Duke University
Financial Technology at Duke University

Version: 1.0

Date: Feb. 11, 2023

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 ranging from 0 to 1. Use PCA and plot the cumulative variance explained by each eigenvalue for $\lambda \in (0, 1)$ each chosen.

What does this tell us about values of and the effect it has on the covariance matrix?

Answer

Model (Exponentially Weighted Volatility):

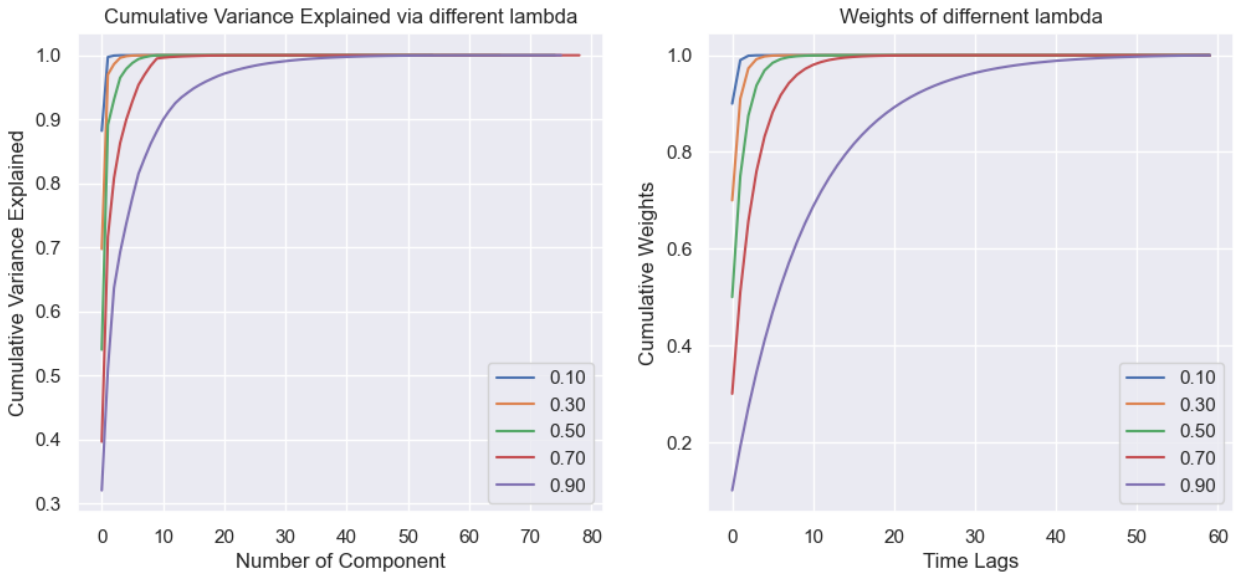
$$\sigma_t^2 = \lambda \sigma_{t-1}^2 + (1-\lambda) r_{t-1}^2$$

Given the data, we choose λ to be 0.1,0.3,0.5,0.7,0.9 and compute the corresponding exponentially weighted covariance matrix of the stock returns of different firms and the weights of different time lags. Then we use PCA to decompose the covariance matrix, which gives us ordered eigenvalues and eigenvectors. Using the results above, we could plot the following images.

We can easily see that as λ increases, the initial value of the cumulative variance explained also increases. With larger λ the whole cumulative variance explained curve moves towards the point(0,1). If we look at the second plot, we can see why this happens. The larger λ, the more evenly the weight of the different time lags is distributed. The smaller $lambda$, the more weight is given to recent stock returns.

Therefore, all of them make sense. A larger λ means that the information is more likely to be evenly distributed. Smaller λ means that recent data contains more information and the information is

concentrated in one place. When we do the PCA, we could use fewer principal components to represent the whole data if λ is large, and vice versa.



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

After we convert the Non-Positive Semi-Definite matrix to a Positive Semi-Definite matrix by Rebonato and Jackel's Method, and Higham's Method, we calculate all eigenvalues of generated a Positive Semi-Definite matrix and find that they are all positive under certain tolerance. Also, we use the previous Cholesky Algorithm to test it and find it's all Positive Semi-Definite matrix.

Given a Non-Positive Semi-Definite matrix, We separately calculate the near Positive Semi-Definite matrix by gradually increasing the matrix size. We use 0,50,100,200 as our matrix size and find that with size grows, the run time of both increases, and the Weighted Frobenius Norm of both also increases. However, Higham's Method spend more time than Rebonato and Jackel's Method and Higham's Method is more precise. Higham's Method consumes more computation power and gets better results.

Therefore, we could safely conclude:

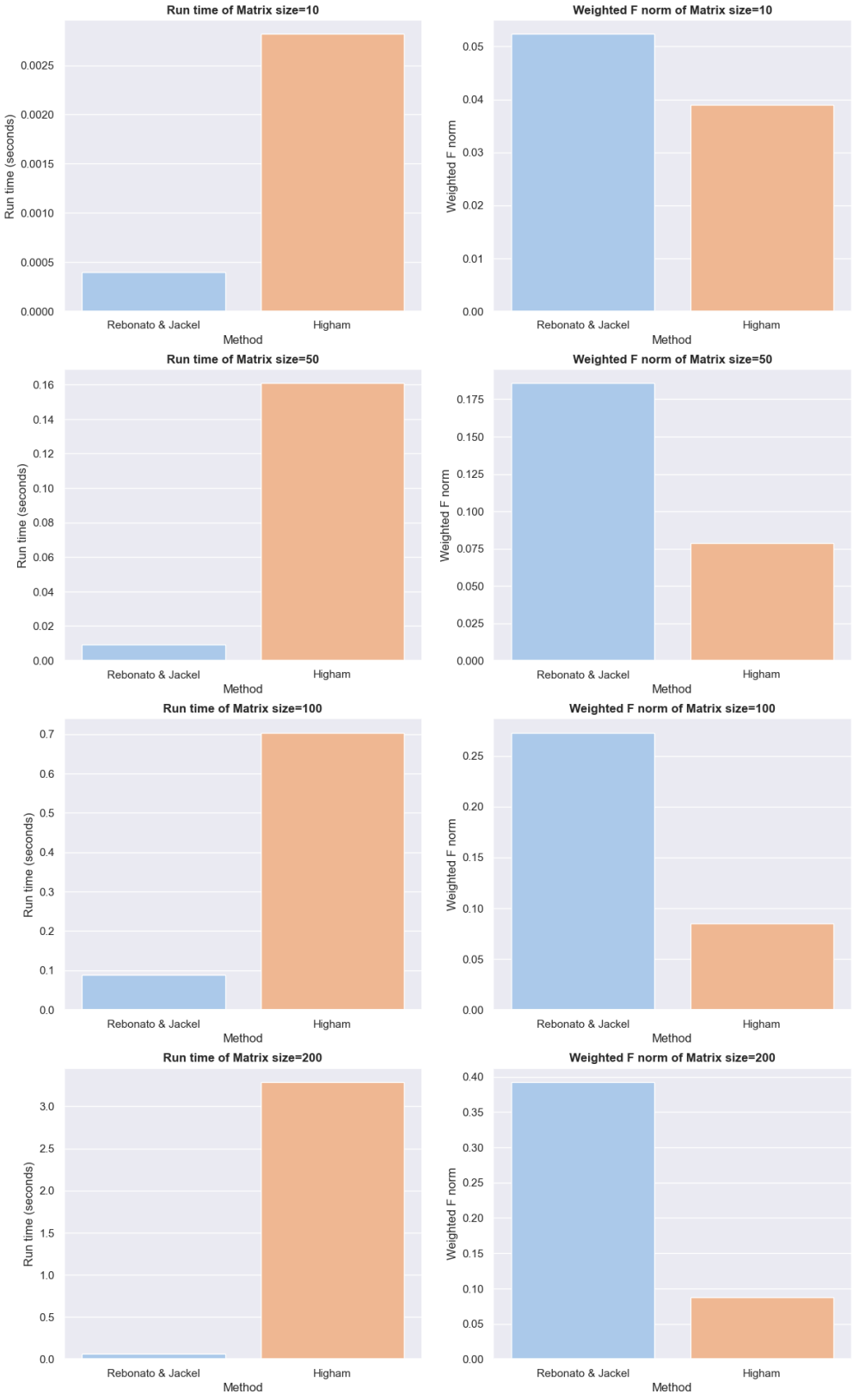
For Rebonato and Jackel's Method:

1. Prons: Fast and Easy to write
2. Cons: The precision is not the smallest

For Higham's Method:

1. Prons: High Precision
2. Cons: Consume more computation resources and comparatively slow

When the matrix size is small, both are fast but the Higham's Method has higher accuracy. Thus I'll choose Higham's Method when the matrix size is not very large. When the matrix size is much bigger, the run time of Higham's Method increases faster and precision maintains. There is a trade-off between time spent and precision got. Therefore, if the time is sufficient and we need high precision of the PSD matrix, I would like to choose Higham's Method. Otherwise, I prefer Rebonato and Jackel's Method.


Run time of Matrix size=10
Run time (seconds)
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
Rebonato & Jackel
Higham
Method
Weighted F norm of Matrix size=10
Weighted F norm
0.00
0.01
0.02
0.03
0.04
0.05
Rebonato & Jackel
Higham
Method
Run time of Matrix size=50
Run time (seconds)
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
0.16
Rebonato & Jackel
Higham
Method
Weighted F norm of Matrix size=50
Weighted F norm
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
Rebonato & Jackel
Higham
Method
Run time of Matrix size=100
Run time (seconds)
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Rebonato & Jackel
Higham
Method
Weighted F norm of Matrix size=100
Weighted F norm
0.00
0.05
0.10
0.15
0.20
0.25
Rebonato & Jackel
Higham
Method
Run time of Matrix size=200
Run time (seconds)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Rebonato & Jackel
Higham
Method
Weighted F norm of Matrix size=200
Weighted F norm
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
Rebonato & Jackel
Higham
Method

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

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

I use all the combinations of EW covariance, EW correlation, covariance, and correlation as input to the simulator. As expected, I find PCA simulation with 100% explained is faster than Direct simulation almost all the time, even though sometimes they are almost the same, just like the following charts. Also, the PCA simulation with 100% explained is almost the same precise as the Direct simulation. Besides, PCA simulation with a lower explanation would be faster but cost its accuracy. We could see from the images that with PCA variance explained percentage decreases, their Weighted Frobenius Norm goes up.

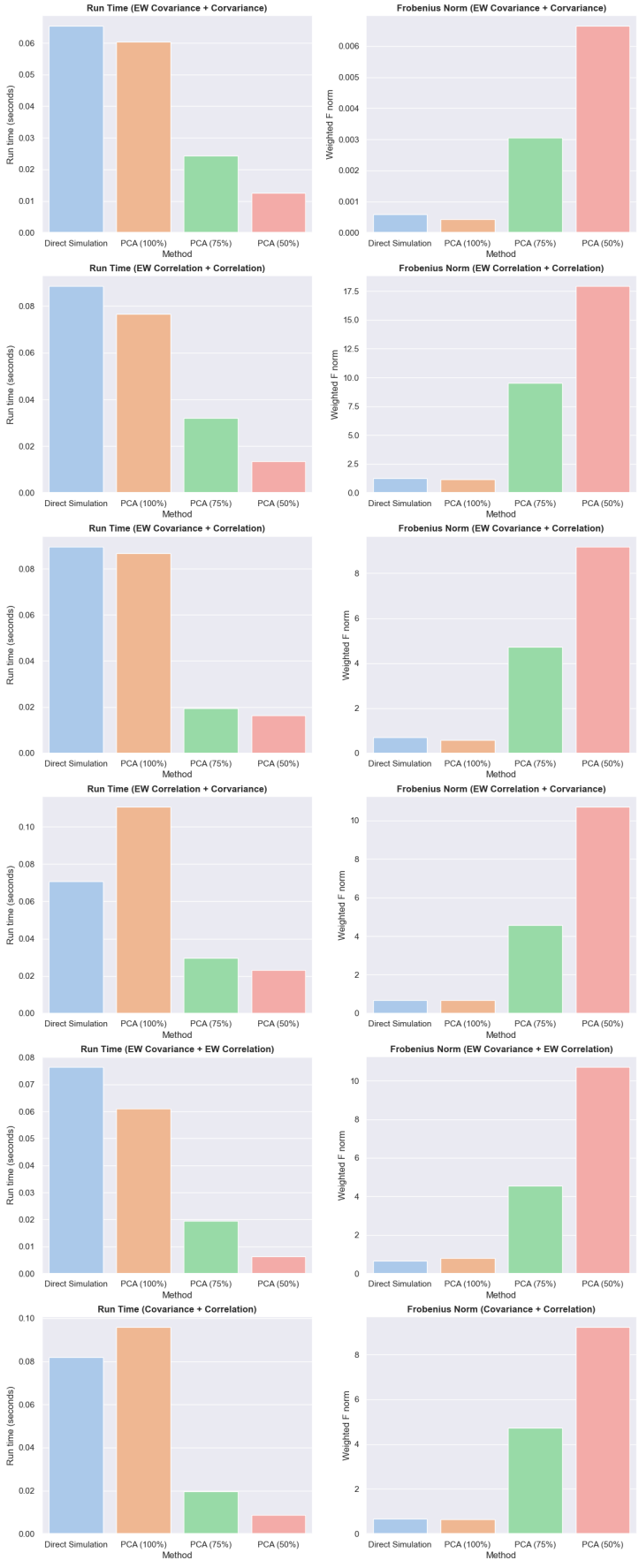
Therefore, such a phenomenon is much the same as Problem 2. If we want to enhance the precision, we need to do more computation to make every digital number as close as the original covariance matrix. Put more time into the calculation, accuracy goes up, and vice versa. When we choose to use which method, it's up to our scenario and tasks.

Table 1: Run Time

Variance	Direct Simulation	PCA (100%)	PCA (75%)	PCA (50%)
EW Covariance + Corvariance	0.06536	0.06028	0.02426	0.01245
EW Correlation + Correlation	0.08846	0.07655	0.03211	0.01337
EW Covariance + Correlation	0.08969	0.08663	0.01932	0.01632
EW Correlation + Corvariance	0.07056	0.11067	0.02962	0.02299
EW Covariance + EW Correlation	0.07644	0.06112	0.01945	0.00627
Covariance + Correlation	0.08181	0.09588	0.01968	0.00859

Table 2: Weighted Frobenius Norm

Variance	Direct Simulation	PCA (100%)	PCA (75%)	PCA (50%)
EW Covariance + Corvariance	0.00058	0.00042	0.00305	0.00665
EW Correlation + Correlation	1.27873	1.14116	9.54698	17.9189
EW Covariance + Correlation	0.69887	0.56596	4.72093	9.19355
EW Correlation + Corvariance	0.6829	0.66773	4.56596	10.71368
EW Covariance + EW Correlation	0.64075	0.80737	4.55341	10.7162
Covariance + Correlation	0.66065	0.63714	4.7269	9.22308


Run Time (EW Covariance + Corvariance)
Run time (seconds)
Direct Simulation
PCA (100%)
PCA (75%)
PCA (50%)
Method
Frobenius Norm (EW Covariance + Corvariance)
Weighted F norm
Run Time (EW Correlation + Correlation)
Frobenius Norm (EW Correlation + Correlation)
Run Time (EW Covariance + Correlation)
Frobenius Norm (EW Covariance + Correlation)
Run Time (EW Correlation + Corvariance)
Frobenius Norm (EW Correlation + Corvariance)
Run Time (EW Covariance + EW Correlation)
Frobenius Norm (EW Covariance + EW Correlation)
Run Time (Covariance + Correlation)
Frobenius Norm (Covariance + Correlation)