**Question 1**

For this question, I wrote a function named “ewm” to create the exponentially weighted covariance matrix. In this function, I just followed the formula in the note. First, I calculate the weight value. Then, I normalize the weight by using the individual weight value divide the total weight value. After, I subtract the each column’s mean for each element in the matrix. Then, I use the dot product to calculate the covariance matrix.

After writing this function, I plug lambda equal to 0.97, 0.7, and 0.85 into the function to see the cumulative variance explained by each eigenvalue for each lambda chosen. The basic math method is very simple. First, I use “eigh” function to calculate the eigenvalue and eigenvector for exponentially weighted covariance matrix. Second, I sort the eigenvalue from large to small. Third, I calculate the total value of the eigenvalue, then using cumulative eigenvalue divide the total value to calculate the percentage explained.

The plot I get is:

图表

描述已自动生成

From the plot, we can see that there needs more eigenvalues involved to explain the data when lambda become larger. Also, we can understand from math equation that W\_(t-i) = (1-lambda)\*lambda^(i-1). In our question, it means that when lambda increases, we are more care about recent data (give more weight for most recent data).

**Question 2**

Followed the instructions of this question, I translate the chol\_psd(), near\_psd(), and higham\_psd() from Julia into Python. Then, I simulate a matrix sigma (500X500) with all 1 on diagonal, and 0.9 for other elements. Also, the first two elements are changed to 0.7357. This is the data that I will used in this question.

Then, I applied the near\_psd() and higham\_psd() on this matrix and record the running time for those two methods. I get the running time for near\_psd() is 0.028581 and higham\_psd() is 0.938344. The higham method will consume more time than near psd method. However, higham will give a nearer psd than near\_psd(). I compare the norm for each method. We can think about the norm as the difference between the original matrix and psd matrix. Higham method gives a smaller Frobenius norm than near\_psd method.

This table is easy to see the difference between nearPSD and Higham2002.

图形用户界面, 文本, 网站

描述已自动生成

**Question 3**

First, I just use corr() and var() to find out the standard pearson correlation matrix and variance values for the data frame. Then, I also have the covariance matrix for lambda = 0.97 from question 1. Then, I use the formula that Cov\_AB = Corr\_AB \* Var\_A \* Var\_B to calculate the correlation matrix for exponentially weight matrix. The Var\_i is the diagonal values for exponentially weight covariance matrix.

Then, now we have pearson’s variance, pearson’s correlation, ewma’s variance, and ewma’s correlation. Applied formula Cov\_AB = Corr\_AB \* Var\_A \* Var\_B, we can get peason’s covariance matrix, ewma’s covariance matrix, covariance matrix combined by pearson’s variance and ewma’s correlation, and covariance matrix combined by pearson’s correlation and ewma’s variance.

Then, I write the normal simulation function and pca simulation function to see the performance of each simulation for those four covariance matrices. I used Frobenius Norm to measure the accuracy, and also I record the running time for each simulation.

Here is the table that I get.

表格

描述已自动生成

We can see that direct simulation give the most accuracy matrix, but it spend more time. With increasing of PCA explained, the norm become smaller, but we need to spend more time to run the code. This is a trade off between the accuracy and spending time.