Instructions:
Be verbose. Explain clearly your reasoning, methods, and results in your written work. Write clear code that is well documented. With 99% certainty, you cannot write too many code comments.

Written answers are worth 8 points. Code is worth 2 points. 10 points total.

1. When finished, respond to the questions in Sakai as "done." We will record your grade there.
2. In your code repository, create a folder called "Week03."
3. In that folder, include
    a. a document (preferably a PDF) with your responses.
    b. Slides for presenting your results.
    c. All code
    d. A README file with instructions for us to run your code

Everything must be checked into your repository by 8am Saturday 1/29. A pull will be done at that time. Documents and code checked in after the instructors pull will not be graded.

You are welcome to use a notebook for code and response documents, but you should not rely on code to explain for you. I would prefer you to use a document editor to write your responses and paste graphs into the document. Use words, math, tables, and charts to explain your results – not code.

This week we will implement the methodologies we discussed in class. These methodologies will be crucial for work later.

Some routines might be available in your programming language, but you need to implement them once to help your understanding. If you rely on a package, you need to prove that it works as expected in this homework. Implementation of these routines will aide in your understanding of the concepts and help you troubleshoot errors later.

You may find that your implementation is faster than your package, or that the package is faster. If you have proved that the package implementation is faster, but and works as needed, you may use it throughout this class.

Data for problems can be found in CSV files with this document in the class repository.

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 λchosen.

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

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.

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.