DS-GA 1002 - Homework 6

Eric Niblock

October 19th, 2020

1. (Correlation coefficient.) For any 2 $\times$ 2 symmetric matrix

$$M = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

such that $b \neq 0$, the eigenvalues of M equal

$$\lambda_1 = \tfrac{a+c+d}{2}$$
$$\lambda_2 = \tfrac{a+c-d}{2}$$

where $d = \sqrt{a^2 + 4b^2 + c^2 - 2ac}$. The eigenvectors equal

$$u_1 = \begin{bmatrix} \frac{a-c+d}{2b} \\ 1 \end{bmatrix}, \qquad u_2 = \begin{bmatrix} \frac{a-c-d}{2b} \\ 1 \end{bmatrix}$$

Note that the eigenvectors are not normalized for simplicity. Use this to prove that for any zero-mean random variables $\tilde{a}$ and $\tilde{b}$ if $\rho_{\tilde{a},\tilde{b}} = 1$ then

$$P\left(\tilde{b} = \frac{\sigma_b}{\sigma_a}\tilde{a}\right) = 1 \tag{1}$$

Let $\tilde{x}$ be a random vector composed of random variables $\tilde{a}$ and $\tilde{b}$. Then, we have that,

$$\Sigma_{\tilde{x}} = \begin{bmatrix} \sigma^2_{\tilde{a}} & Cov(\tilde{a},\tilde{b}) \\ Cov(\tilde{a},\tilde{b}) & \sigma^2_{\tilde{b}} \end{bmatrix} \tag{2}$$

Now since we have $\rho_{\tilde{a},\tilde{b}} = 1$, this implies that $Cov(\tilde{a},\tilde{b}) = \sigma_{\tilde{a}}\sigma_{\tilde{b}}$. The covariance matrix of $\tilde{x}$ then becomes,

$$\Sigma_{\tilde{x}} = \begin{bmatrix} \sigma_{\tilde{a}}^2 & \sigma_{\tilde{a}}\sigma_{\tilde{b}} \\ \sigma_{\tilde{a}}\sigma_{\tilde{b}} & \sigma_{\tilde{b}}^2 \end{bmatrix} \tag{3}$$

We then can decompose the covariance matrix via spectral decomposition for symmetric matrices, yielding the result,

$$\Sigma_{\tilde{x}} = \begin{bmatrix} \frac{\sigma_{\tilde{a}}}{\sigma_{\tilde{b}}} & -\frac{\sigma_{\tilde{b}}}{\sigma_{\tilde{a}}} \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \sigma_{\tilde{a}}^2 + \sigma_{\tilde{b}}^2 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sigma_{\tilde{a}}}{\sigma_{\tilde{b}}} & 1 \\ -\frac{\sigma_{\tilde{b}}}{\sigma_{\tilde{a}}} & 1 \end{bmatrix} \tag{4}$$

Where the central matrix is diagonal, and formed from eigenvalues λ_1 and λ_2. The left matrix is composed of eigenvectors u_1 and u_2, and the right matrix is the transpose of the left matrix. We want to prove a relationship involving the probability of $\tilde{a}$ and $\tilde{b}$. Let's define a new random variable as follows: $\tilde{y} = x_1\tilde{a} + x_2\tilde{b}$. Since $\tilde{a}$ and $\tilde{b}$ have an expected value of 0, it follows readily that $E[\tilde{y}] = 0$. Then, by Chebyshev's inequality, we see that,

$$P(|\tilde{y} - E[\tilde{y}]| \geq c) \leq \frac{Var(\tilde{y})}{c} \tag{5}$$

$$P(|\tilde{y}| \geq c) \leq \frac{Var(\tilde{y})}{c} \tag{6}$$

If we have $Var(\tilde{y}) = 0$, then the probability of $\tilde{y}$ being greater than any positive constant becomes zero. Therefore we find,

$$Var(\tilde{y}) = 0 \implies P(\tilde{y} = 0) = 1 \tag{7}$$

From our decomposition, we find that λ_2 corresponds to a variance of 0, so x_1 and x_2 correspond to the values of the second eigenvector. Then we have,

$$x_1 = -\frac{\sigma_b}{\sigma_a}, \qquad x_2 = 1 \tag{8}$$

$$P(\tilde{y} = 0) = P(-\frac{\sigma_b}{\sigma_a}\tilde{a} + \tilde{b}) = P(\tilde{b} = \frac{\sigma_b}{\sigma_a}\tilde{a}) = 1 \tag{9}$$

Thus, we have proven the desired relation, given that $\rho_{\tilde{a},\tilde{b}} = 1$.

2. (Financial data) In this exercise you will use the code in the *findata* folder. For the data loading code to work properly, make sure you have the pandas Python package installed on your system.

Throughout, we will be using the data obtained by calling *load_data()* in *findata_tools.py*. This will give you the names, and closing prices for a set of 18 stocks over a period of 433 days ordered chronologically. For a fixed stock (such as msft), let $P_1, ..., P_{433}$ denote its sequence of closing prices ordered in time. For that stock, define the daily returns series $R_i := P_{i+1} - P_i$ for $i = 1, ..., 432$. Throughout we think of the daily stock returns as features, and each day (but the last) as a separate datapoint in $\mathbb{R}^{18}$. That is, we have 432 datapoints each having 18 features.

(a) Looking at the first two principal directions of the centered data, give the two stocks with the largest coefficients (in absolute value) in each direction. Give a hypothesis why these two stocks have the largest coefficients, and confirm your hypothesis using the data. The file *findata_tools.py* has *pretty_print()* functions that can help you output your results. You are not required to include the principal directions in your submission.

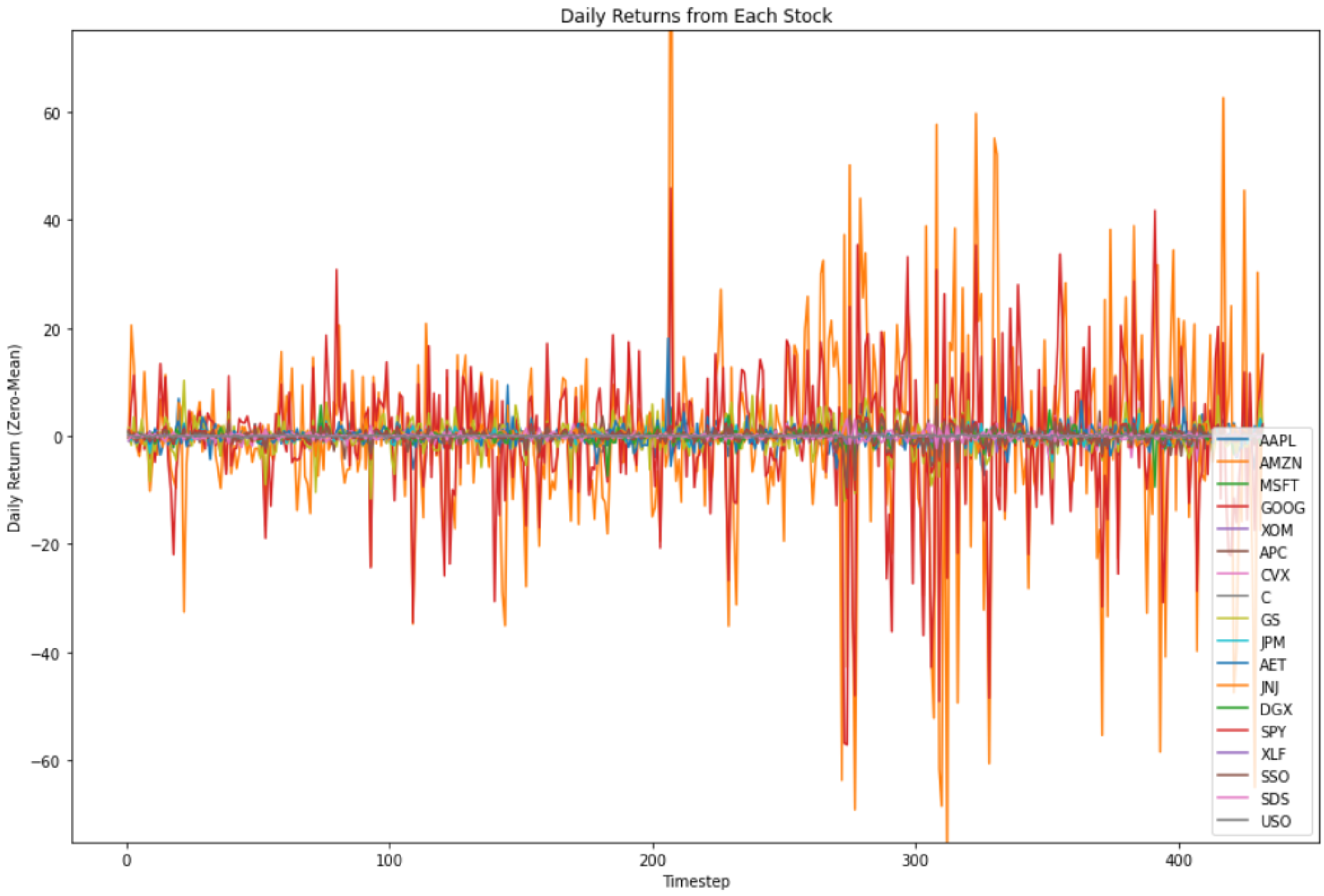
We can observe which two features (or stocks) have the largest coefficients in each of the first two directions by constructing the returns matrix in then preforming PCA on the zero-mean returns matrix. Observe the following code,

```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA

def load_data(filename) :
    df = pd.read_csv(filename,index_col=0)
    names = df.columns.values.tolist()
    data = df.values
    return names,data
names, data = load_data(r'stockprices.csv')

def pretty_print(vec,names) :
    print(pd.DataFrame(vec,names,['']).transpose())
```

```
## Construction of Returns Matrix
shifted_data = data
shifted_data = np.delete(shifted_data, 0, 0)
shifted_data = np.vstack((shifted_data, data[-1,:]))
return_matrix = shifted_data - data
return_matrix = return_matrix[0:-1,:]

##Scaling Returns Matrix to Zero Mean
mean_zero = StandardScaler(with_std=False).fit_transform(return_matrix)

##Performing PCA on Zero-Mean Data
pca = PCA()
pca.fit(mean_zero)
print('First Eigenvector (PC1)')
print('  ')
pretty_print(abs(pca.components_[0]),names)
print('  ')
print('Second Eigenvector (PC2)')
print('  ')
pretty_print(abs(pca.components_[1]),names)
```

```
First Eigenvector (PC1)

     AAPL     AMZN      MSFT      GOOG       XOM       APC       CVX  \
 0.054553  0.86793  0.036651  0.482672  0.007916  0.009795  0.013876

        C        GS       JPM       AET       JNJ       DGX       SPY  \
 0.012407  0.053403  0.020728  0.008575  0.013319  0.011993  0.054358

      XLF       SSO       SDS       USO
 0.004979  0.044236  0.016887  0.001485

Second Eigenvector (PC2)

     AAPL      AMZN      MSFT      GOOG       XOM       APC      CVX  \
 0.041894  0.494995  0.026756  0.851087  0.028004  0.009165  0.02881

       C        GS       JPM       AET       JNJ       DGX       SPY  \
 0.02591  0.114993  0.037115  0.028005  0.041139  0.011263  0.069472

      XLF       SSO       SDS       USO
 0.009083  0.056721  0.021421  0.000401
```

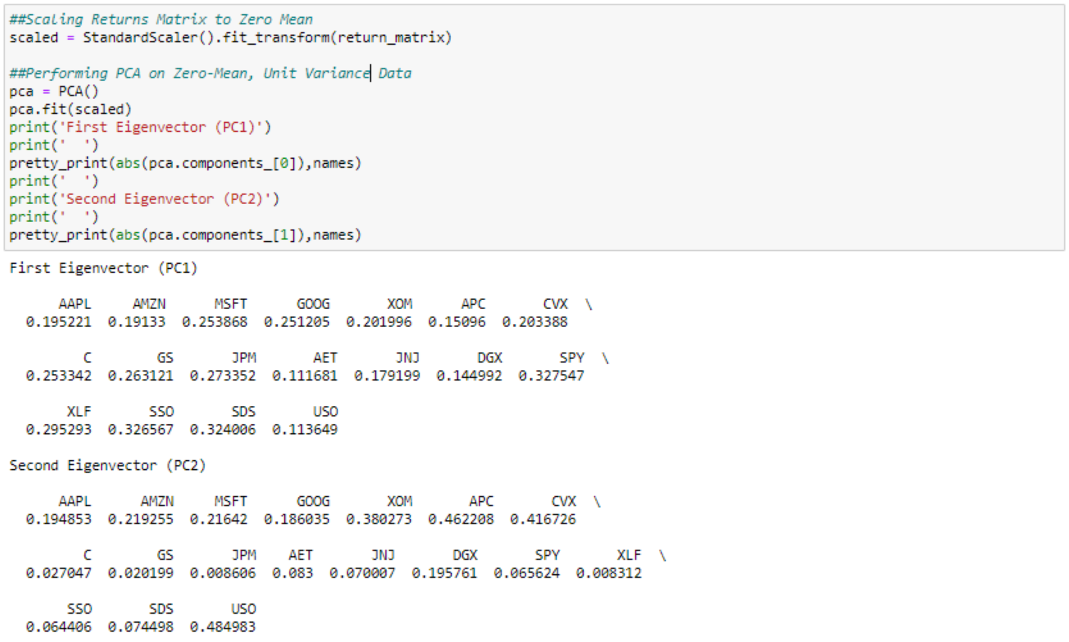
From the first two eigenvectors, it becomes apparent that concerning eigenvector one, Amazon has the largest component along it's direction. Concerning eigenvector two, Google appears to have the largest eigenvector along it's direction. If we observe the daily return of each stock in a chart, it then becomes clear that these two companies are those with the largest amount of variance.



It is evident that these two stocks, Amazon and Google, represented by the orange and red lines respectively, have the largest fluctuation (variance) which corresponds to having large components within the first two principal directions.

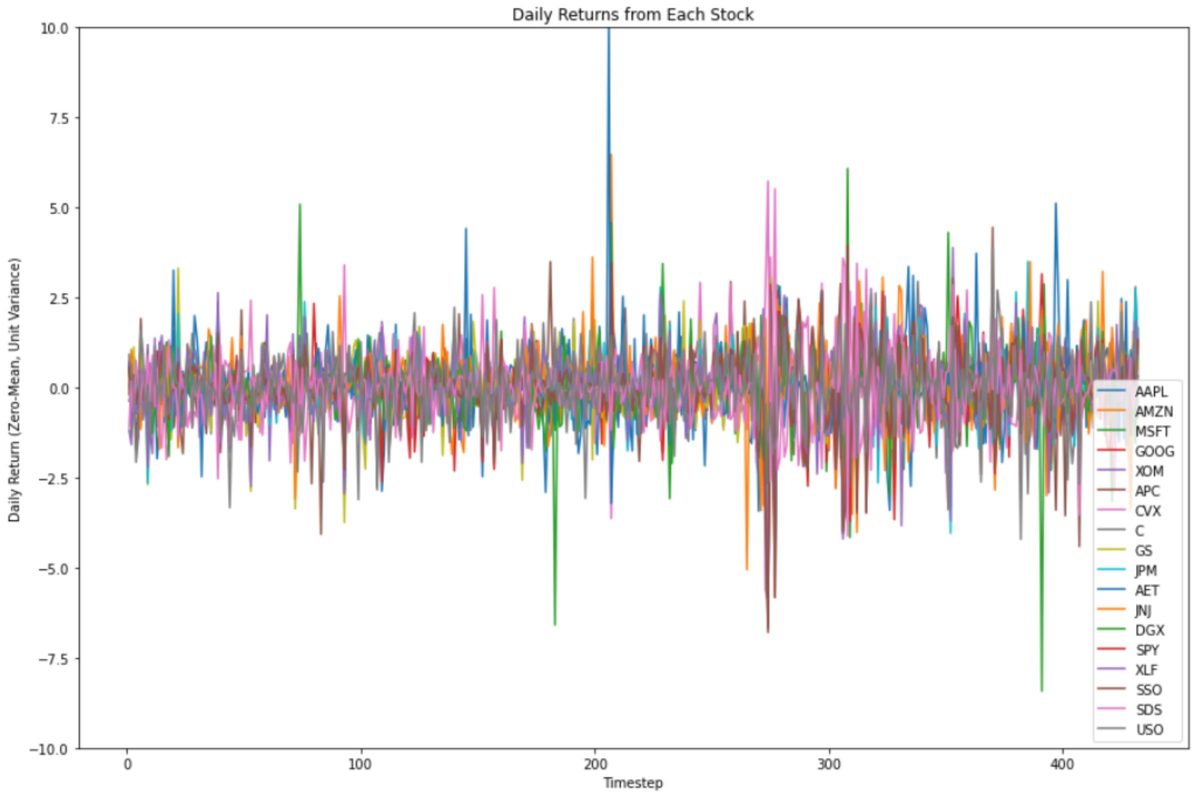
(b) Standardize the centered data so that each stock (feature) has variance 1 and compute the first 2 principal directions. This is equivalent to computing the principal directions of the correlation matrix (the previous part used the covariance matrix). Using the information in the comments of generate findata.py as a guide to the stocks, give an English interpretation of the first 2 principal directions computed here. You are not required to include the principal directions in your submission.

We perform the same PCA, though this time we center the data and also scale the data such that each feature vector has unit variance. The results are shown below,



```
##Scaling Returns Matrix to Zero Mean
scaled = StandardScaler().fit_transform(return_matrix)

##Performing PCA on Zero-Mean, Unit Variance Data
pca = PCA()
pca.fit(scaled)
print('First Eigenvector (PC1)')
print('  ')
pretty_print(abs(pca.components_[0]),names)
print('  ')
print('Second Eigenvector (PC2)')
print('  ')
pretty_print(abs(pca.components_[1]),names)
```

```
First Eigenvector (PC1)

      AAPL     AMZN     MSFT     GOOG      XOM      APC      CVX  \
  0.195221  0.19133  0.253868  0.251205  0.201996  0.15096  0.203388

         C       GS      JPM      AET      JNJ      DGX      SPY  \
  0.253342  0.263121  0.273352  0.111681  0.179199  0.144992  0.327547

       XLF      SSO      SDS      USO
  0.295293  0.326567  0.324006  0.113649

Second Eigenvector (PC2)

      AAPL     AMZN     MSFT     GOOG      XOM      APC      CVX  \
  0.194853  0.219255  0.21642  0.186035  0.380273  0.462208  0.416726

         C       GS      JPM    AET      JNJ      DGX      SPY      XLF  \
  0.027047  0.020199  0.008606  0.083  0.070007  0.195761  0.065624  0.008312

       SSO      SDS      USO
  0.064406  0.074498  0.484983
```



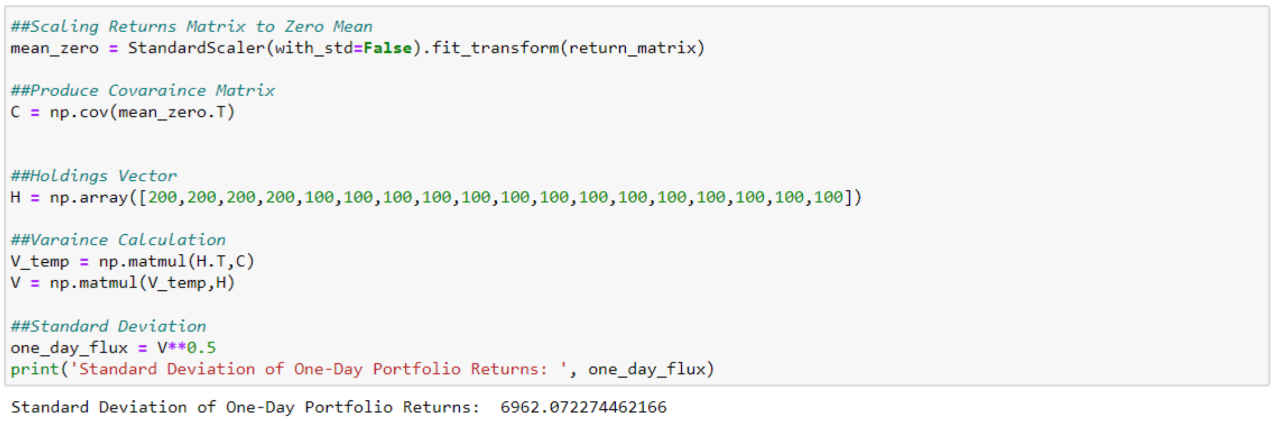
Here, we find that the two stocks with the largest coefficient have changed. In regards to the first eigenvector, we have State Street's SPDR S&P 500 ETF (SPY), and in regards to the second eigenvector, we have, USO, an exchange traded product that tracks the price of oil in the US. This result makes sense, and can be interpreted as follows. When using the covariance matrix for PCA, the stocks with the highest variance dominated our analysis. When using the correlation matrix, the variances were all set to unit length, which allows PCA to produce a better intuition relying more heavily on the relationship between the features (stocks), as opposed to simply evaluating their variances. SPY and USO both are most reliable in terms of representing the other features (the S&P 500 represents the top 500 companies, many of which are listed here, and oil is generally a good barameter of market conditions), and therefore they dominate the first two principle directions.

(c) **Assume the stock returns each day are drawn independently from a multivariate distribution $\tilde{x}$ where $\tilde{x}[i]$ corresponds to the $i-$th stock. Assume further that you hold a portfolio with 200 shares of each of appl, amzn, msft, and goog, and 100 shares of each of the remaining 14 stocks in the dataset. Using the sample covariance matrix as an estimator for the true covariance of $\tilde{x}$, approximate the standard deviation of your 1 day portfolio returns $\tilde{y}$ (this is a measure of the risk of your portfolio). Here $\tilde{y}$ is given by**

$$\tilde{y} = \textstyle\sum_{i=1}^{18} \alpha[i]\tilde{x}[i]$$

where $\alpha[i]$ is the number of shares you hold of stock i.

The following section of code provides the standard deviation of our daily earnings given the described holdings,



```
##Scaling Returns Matrix to Zero Mean
mean_zero = StandardScaler(with_std=False).fit_transform(return_matrix)

##Produce Covaraince Matrix
C = np.cov(mean_zero.T)


##Holdings Vector
H = np.array([200,200,200,200,100,100,100,100,100,100,100,100,100,100,100,100,100,100])

##Varaince Calculation
V_temp = np.matmul(H.T,C)
V = np.matmul(V_temp,H)

##Standard Deviation
one_day_flux = V**0.5
print('Standard Deviation of One-Day Portfolio Returns: ', one_day_flux)
```

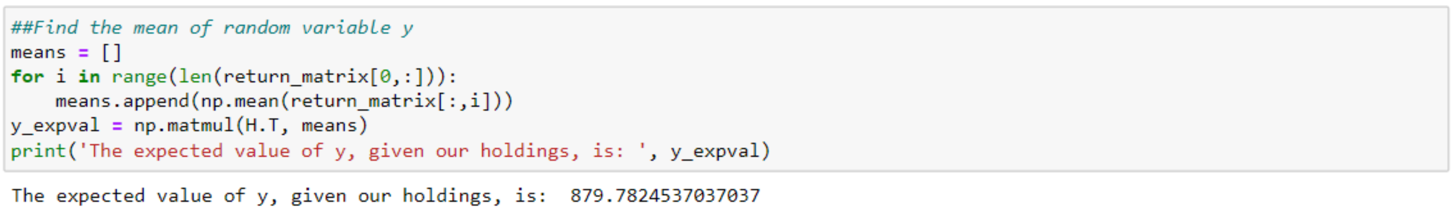
```
Standard Deviation of One-Day Portfolio Returns:  6962.072274462166
```

So the standard deviation of our daily return given our holdings is $\sigma_{\tilde{y}} \approx 6962$.

(d) **Assume further that $\tilde{x}$ from the previous part has a multivariate Gaussian distribution. Compute the probability of losing 1000 or more dollars in a single day. That is, compute**

$$Pr(\tilde{y} \leq -1000)$$

We will model the probability of losing more than 1000 dollars within a single day by using a Gaussian random variable. We therefore must first calculate the expected value of $\tilde{y}$, shown in the code below,



```
##Find the mean of random variable y
means = []
for i in range(len(return_matrix[0,:])):
    means.append(np.mean(return_matrix[:,i]))
y_expval = np.matmul(H.T, means)
print('The expected value of y, given our holdings, is: ', y_expval)
```

```
The expected value of y, given our holdings, is:  879.7824537037037
```

Thus, $\mu_{\tilde{y}} \approx 880$. So, given, $\mu_{\tilde{y}}$ and $\sigma_{\tilde{y}}$, we can use a normal table to calculate that $Pr(\tilde{y} \leq -1000) \approx 0.3936$.

3. (Streaks) In this problem we consider the problem of testing whether a randomly generated sequence is truly random. A certain computer program is supposed to generate Bernoulli iid sequences with parameter 0.5. When you try it out, you are surprised that it contains long streaks of 1s. In particular, you generate a sequence of length 200, which turns out to contain a sequence of 8 ones in a row.

(a) Let $\tilde{s}$ be equal to the longest streak of 1s in an iid Bernoulli sequence of length 5. Compute the pmf of $\tilde{s}$ exactly.

Since the sequence is only of length 5, there is only $2^5 = 32$ possible outcomes. The easiest method of generating the pmf is by analyzing the sample space, given below,

$$\begin{array}{cccccc}
00000 & 00001 & 00011 & 11100 & 11110 & 11111 \\
 & 00010 & 00101 & 11010 & 11101 & \\
 & 00100 & 01001 & 10110 & 11011 & \\
 & 01000 & 10001 & 01110 & 10111 & \\
 & 10000 & 11000 & 00111 & 01111 & \\
 & & 10100 & 01011 & & \\
 & & 10010 & 01101 & & \\
 & & 01100 & 10011 & & \\
 & & 00110 & 11001 & & \\
 & & 01010 & 10101 & &
\end{array} \tag{10}$$

So, now we simply note the longest streak within each possible entry of the sample space,

$$\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
 & 1 & 1 & 2 & 3 & \\
 & 1 & 1 & 2 & 2 & \\
 & 1 & 1 & 3 & 3 & \\
 & 1 & 2 & 3 & 4 & \\
 & & 1 & 2 & & \\
 & & 1 & 2 & & \\
 & & 2 & 2 & & \\
 & & 2 & 2 & & \\
 & & 1 & 1 & &
\end{array} \tag{11}$$

Then, counting the frequency of each chain length and dividing by 32 yields the pmf,

$$
\begin{aligned}
p_{\tilde{s}}(0) &= {}^{1}\!/_{32} \\
p_{\tilde{s}}(1) &= {}^{12}\!/_{32} \\
p_{\tilde{s}}(2) &= {}^{11}\!/_{32} \\
p_{\tilde{s}}(3) &= {}^{5}\!/_{32} \\
p_{\tilde{s}}(4) &= {}^{2}\!/_{32} \\
p_{\tilde{s}}(5) &= {}^{1}\!/_{32}
\end{aligned}
\tag{12}
$$

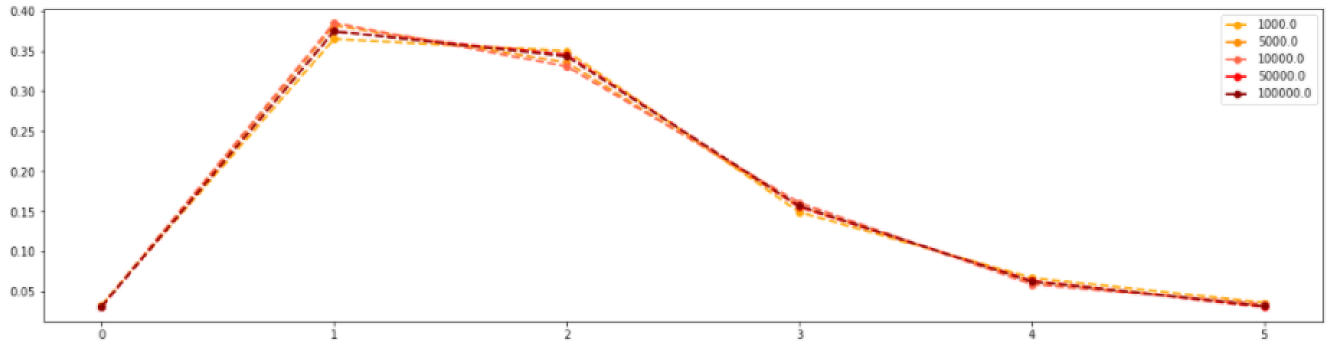
This is the requested pmf of $\tilde{s}$.

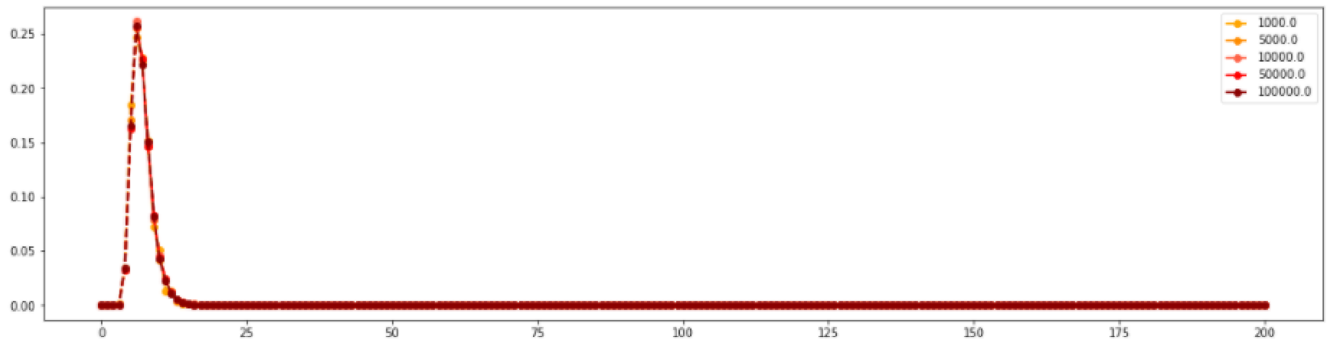
(b) Complete the script *streaks.py* to estimate the pmf of $\tilde{s}$ using Monte Carlo simulation.Compare it to your answer in the previous question. The script will also apply your code to estimate the pmf of $\tilde{s}$ when the Bernoulli iid sequence has length 200. Include your code in the answer as well as the figures generated by the script.

Below is my code and the resulting graphs from the completed script *streaks.py.* We note that the results our computed pmf of $\tilde{s}$ match the Monte Carlo results shown in the first graph.

```python
import numpy as np
import matplotlib.pyplot as plt
plt.close("all")
np.random.seed(2017)

def max_runs_ones(b):
    maxval=0
    for b, g in itertools.groupby(b):
        if b:
            maxval=max(maxval,sum(g))
    return maxval

def pmf_longest_streak(n, tries):
    pmf = np.zeros(n+1)

    for t in range(tries):
        val = max_runs_ones(np.array([np.random.randint(0,2) for i in range(n)])>0)
        pmf[val] += 1
    return(pmf/tries)

n_tries = [1e3,5e3,1e4,5e4,1e5]

n_vals = [5,200]

color_array = ['orange','darkorange','tomato','red', 'darkred', 'tomato', 'purple', 'grey', 'deepskyblue',
               'maroon','darkgray','darkorange', 'steelblue', 'forestgreen', 'silver']
pmfs = []
for ind_n in range(len(n_vals)):
    n = n_vals[ind_n]
    plt.figure(figsize=(20,5))
    for ind_tries in range(len(n_tries)):
        tries = n_tries[ind_tries]
        print ("n: " + str(n) + "   " + "tries: " + str(tries))
        pmf_longest_tries = pmf_longest_streak(n, np.int(tries))
        ### Save pmfs
        pmfs.append(pmf_longest_tries)
        ###
        plt.plot(range(n+1),pmf_longest_tries, marker='o',markersize=6,linestyle="dashed",lw=2,
                 color=color_array[ind_tries],
                 markeredgecolor= color_array[ind_tries],label=str(tries))
    plt.legend()

print ("The probability that the longest streak of ones in a Bernoulli iid sequence of length 200 has length 8 or more is ")
print (sum(pmfs[-1][8:]))
```

```
n: 5   tries: 1000.0
n: 5   tries: 5000.0
n: 5   tries: 10000.0
n: 5   tries: 50000.0
n: 5   tries: 100000.0
n: 200   tries: 1000.0
n: 200   tries: 5000.0
n: 200   tries: 10000.0
n: 200   tries: 50000.0
n: 200   tries: 100000.0
The probability that the longest streak of ones in a Bernoulli iid sequence of length 200 has length 8 or more is
0.31989000000000006
```





(c) Approximate the probability that the longest streak of ones in a Bernoulli iid sequence of length 200 has length 8 or more. Is the sequence of 8 ones evidence that the program may not be generating truly random sequences?

We find, from our empirical Monte Carlo simulation, that the probability of the longest streak of ones in a Bernoulli iid sequence of length 200 has length 8 or more is approximately 0.320 (as shown above). This relatively large sequence of ones is *not* evidence that the program is failing to generate completely random sequences. In fact, human perception is generally very biased in terms of evaluating randomness. The ”gambler’s fallacy” is the belief that increasing runs of one outcome (in this case, ones) makes the other outcome more likely. This is obviously erroneous.

4. **(Radioactive sample) Consider the following experiment. We have a radioactive sample situated at unit distance from a line of sensors. Each time a sensor detects a particle emitted from the sample we obtain a reading of the position of the sensor in the x axis (we assume that we have so many sensors that you can model this position as a continuous random variable). We model the measurements as an i.i.d. sequence distributed as a random variable $\tilde{m} = c + \tilde{x}$ where the pdf of $\tilde{x}$ is symmetric around the origin, that is $f_{\tilde{x}}(x) = f_{\tilde{x}}(-x)$ for all real numbers x. Your task is to estimate the position of the sample c from these data.**

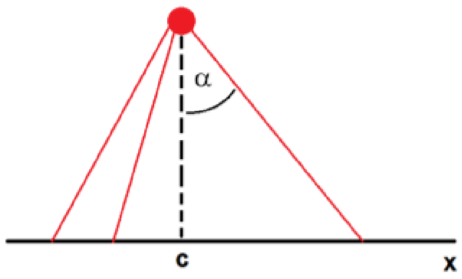
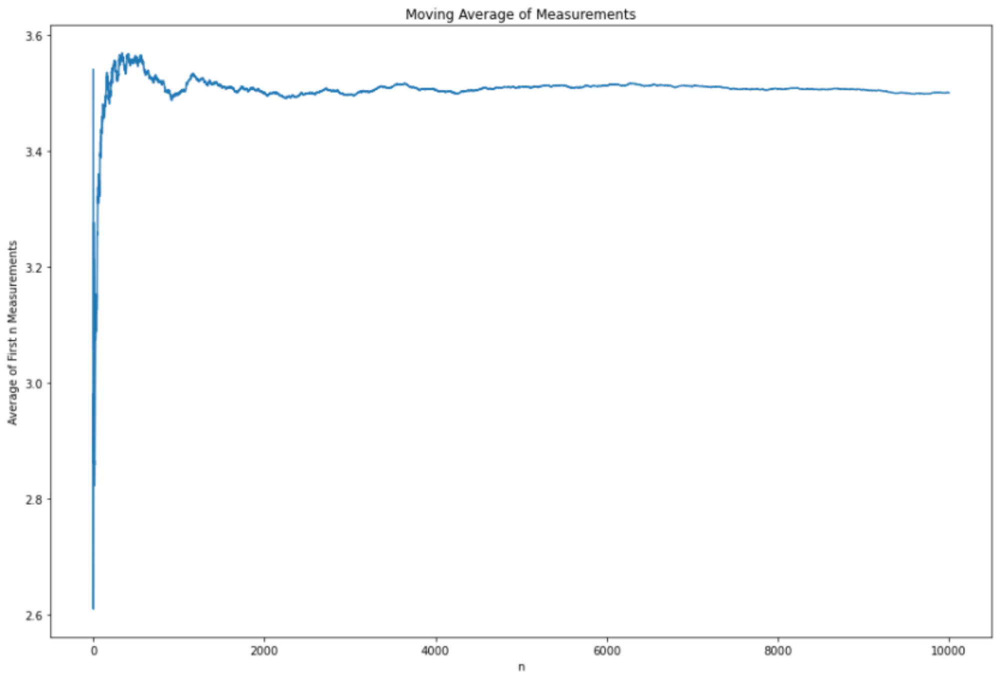


Figure 1: Diagram of the experiment.

   (a) **The file *radioactive_sample_1.txt* contains a vector of measurements $m_1, m_2, \ldots$. Plot a moving average of the measurements $\frac{1}{n}\sum_{i=1}^{n} m_i$ for $n = 1, 2, 3, \ldots$ (and submit the plot). Use the plot to give an estimate for the value of c. (Hint: What is the expected value of $\tilde{m}$?)**

   Under what assumptions on $\tilde{x}$ can you prove that the estimation method you propose work?

   Below is the plot of moving averages generated from the data in *radioactive_sample_1.txt*,



The final moving average, which is simply the average of all of the measurements within *radioactive_sample_1.txt*, is given as approximately 3.501. We also know that,

$$\tilde{m} = c + \tilde{x} \tag{13}$$

$$\begin{aligned} E[\tilde{m}] &= E[c + \tilde{x}] \\ E[\tilde{m}] &= E[c] + E[\tilde{x}] \qquad (14) \\ E[\tilde{m}] &= E[c] = c \end{aligned}$$

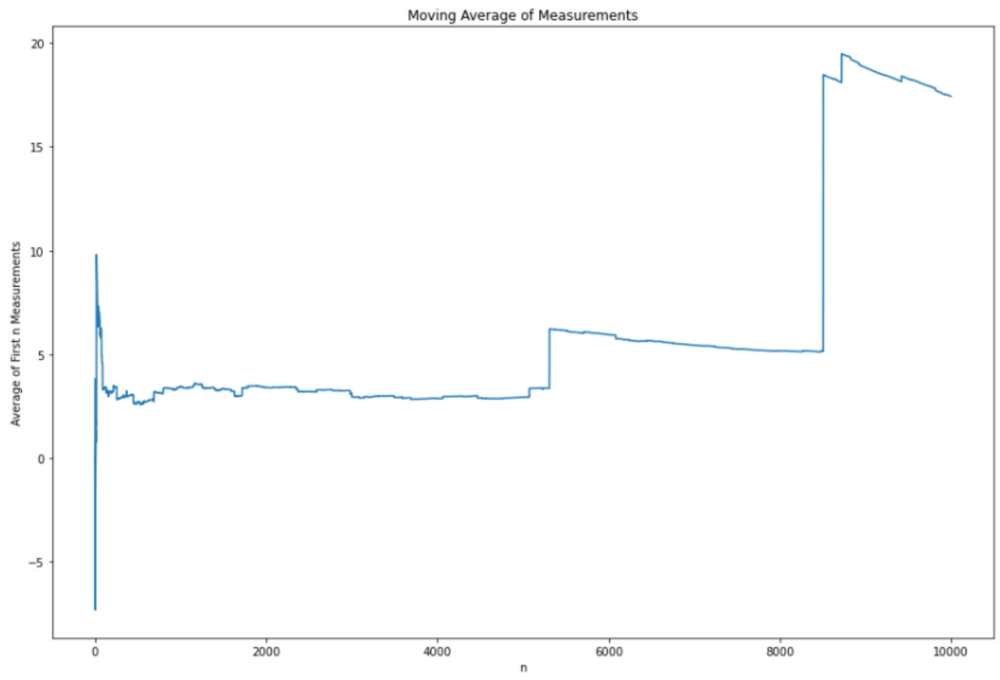
Since the law of large numbers implies that the arithmetic mean of the measurements converges to the expected value of $\tilde{m}$, we can take $E[\tilde{m}] = 3.501$. Therefore,

$$c = E[\tilde{m}] \approx 3.501 \tag{15}$$

In order to show that this estimation method is valid, it must be the case that $E[\tilde{x}] = 0$. This is a reasonable assumption, given the problem statement, since the pdf of $\tilde{x}$ is symmetric around $x = 0$. Therefore, we only make the assumption that the expected value is well-defined.

(b) **The file *radioactive_sample_2.txt* contains a vector of measurements corresponding to a different radioactive sample. Does the estimation method described above work in this case? Submit the plot of the new moving average.**

The estimation method does not work well here. The plot of the moving average is as follows,



We can clearly see that the expected value of the measurements does not converge given a large value of n.

(c) **A colleague suggests that the angle α between the trajectory of the particles emitted by the new sample and the vertical axis (illustrated in Figure 1) might be well modeled by a random variable $\tilde{a}$ that is uniformly distributed between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$. Compute the pdf and mean of $\tilde{x}$ under this assumption. (Hint: remember the trigonometric function tan and its inverse arctan.) Would such model explain your observations in (b)?**

If we model α as uniform between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$, we have the following pdf,

$$f_{\tilde{a}}(a) = \begin{cases} \frac{1}{\pi}, & \text{if } -\frac{\pi}{2} \leq a \leq \frac{\pi}{2} \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

We also know that the relationship between $\tilde{a}$ and $\tilde{x}$ is given by,

$$\tilde{x} = tan(\tilde{a}) \tag{17}$$

Then, we have that,

$$\begin{aligned} F_{\tilde{x}}(x) &= P(\tilde{x} \leq x) = P(tan(\tilde{a}) \leq x) = P(\tilde{a} \leq arctan(x)) \\ &= F_{\tilde{a}}(arctan(x)) \end{aligned} \tag{18}$$

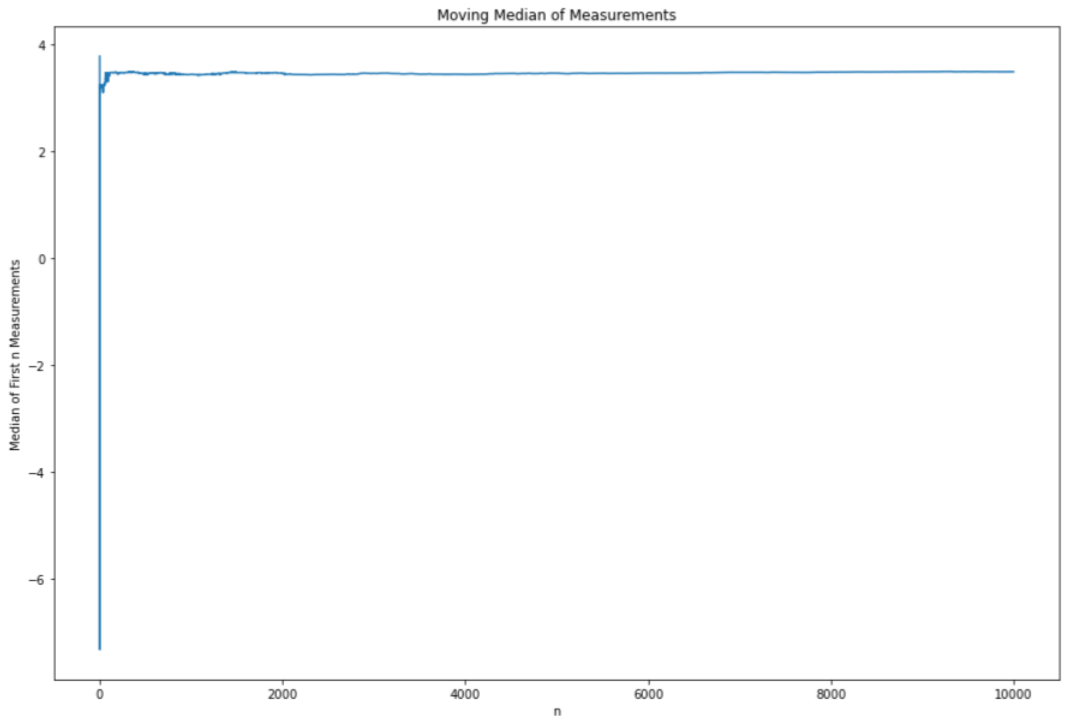
And the derivative of both sides yields,

$$f_{\tilde{x}}(x) = \frac{1}{1+x^2} f_{\tilde{a}}(arctan(x)) = \frac{1}{\pi(1+x^2)} \tag{19}$$

This confirms the results we found in part (b) because the resulting pdf of $\tilde{x}$ is a Cauchy random variable, which has an undefined expected value. This explains why the expected value of the measurements in part (b) did not converge.

(d) The sample mean can be affected by extreme values and outliers, whereas the sample median is more robust. The sample median converges to the median of an iid sequence of random variables even when the mean is not well defined. Use the sample median from *radioactive_sample_2.txt* to estimate c.

The result is simply that $c = median(\tilde{m})$. Observe the following plot, which depicts the moving median,



So, we have that,

$$c = median(\tilde{m}) \approx 3.495 \tag{20}$$