ENGG*6405: Finance and Economics for Engineers

Monthly Returns

Amir A. Aliabadi

This document is typeset using LaTeX

April 9, 2025

1 Introduction

`Python` is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, `Python` has a design philosophy which emphasizes code readability (notably using white space indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

`Python` features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

`Python` interpreters are available for many operating systems, allowing `Python` code to run on a wide variety of systems. `CPython`, the reference implementation of `Python`, is open source software and has a community-based development model, as do nearly all of its variant implementations. `CPython` is managed by the non-profit `Python` Software Foundation.

2 Installation

To install `Python`, go to the following link and download the latest version for the appropriate operating system. Complete the installation steps.

`https://www.python.org/downloads/`

The standard application launcher for `Python` is `IDLE`, which opens a `Shell`. A shell is a user interface for access to an operating system's services. In general, operating system shells use either a command-line interface (CLI) or graphical user interface (GUI), depending on a computer's role and particular operation. It is named a shell because it is a layer around the operating system kernel. `IDLE` is a CLI shell that is shown in Figure 2.

Figure 1: The logo of Python programming language.

Another application launcher for Python is IDE from PyCharm that can be downloaded from the following link. Download the Community version of IDE from PyCharm, which is lightweight and suitable for scientific development, for the appropriate operating system.

http://www.jetbrains.com/pycharm/download/

By launching IDE from PyCharm the following window opens that asks to open a new project or an existing project, as shown in Figure 3. Chose to open a new project and specify the directory path for the project files to be developed.

3 Creating and Running a Simple Program

Click menu item File and then New to create a Python text file. Name the file PythonProgramming. To run a simple python program to print Hello World! in the output console, enter the following lines of code within the editor just created for PythonProgramming.

```
import random
import sys
import os

#This line prints a message
print("Hello World!")
```

The import command allows us to use modules from various libraries in order to perform specific programming tasks. The random module allows us to generate a random number. The sys and os modules launch the operating system. The line #This line prints a message is a comment that is not going to be executed. The print("...") command prints a message on the output console. In Python programming both double quotations " and single quotations ' perform the same task. For instance we could have used print('...') in the above program.

Then click menu item Run and then click Run. A run console sub window opens within the IDE where the message Hello World! will be printed, as shown in Figure 4.

Figure 2: An example of the `IDLE` shell that is accessible by standard installation of `Python` on Mac operating system.

4 Installing Python Interpreter Packages

`Python` programming is based on numerous interpreter packages that have been developed by the community. The appropriate packages for a `Python` program must be installed before a package can be used using the `import` command. To install an interpreter package the menu item `PyCharm Community Edition` should be used and then `Preferences` must be clicked. When the `Preferences` window opens, the `Project Interpreter` under the current project can be opened and viewed, as shown in Figure 5.

The list of all interpreter packages can be viewed by double-clicking on a package. A list of all interpreters sorted alphabetically can be obtained. It is possible to search for an interpreter by starting to type the interpreter name as shown in Figure 6. For instance, information about the `numpy` interpreter is shown in the `Available Packages` window. A brief description is provided on the right. This interpreter is used for array processing for numbers, strings, records, and objects. The `Version` and interpreter `Author` is also provided. It is possible to install the interpreter

Figure 3: Openning screen of the `IDE` launcher.

package by selecting the desired version and then check marking the `Install` option. Finally the botton `Install Package` can be clicked. After installation, this package or module can be used by the `import` command in the `Python` program.

5 Monthly Returns

The *Chi-squared test* is used to determine if sample data set is consistent with an expected theoretical distribution, e.g. normal or student's t distributions. If we take a sample of n_b data points for which we can compute the expected values according to a distribution (e.g. normal), then we can compare how well the sampled values agree with the expected distribution. The Chi-squared can be calculated for the comparison as

$$\chi^2 = \sum_{i=1}^{n_b} \left[\frac{\text{Sample Values - Expected Values}}{\text{Standard Deviation}} \right]^2 , \tag{1}$$

where the standard deviation comes from the sample and $n_b \neq n$ is the number of bins for the comparison. The χ^2 value is a reasonable indicator of the agreement. If $\chi^2 = 0$, then the expected distribution and sample match perfectly. The larger the χ^2 the lower the probability that the sample values match the expected distribution. To conduct this test, the sample values are divided into n_b bins. Each bin must contain at least one sample data point. We compute the width of each bin using $\Delta r = r_{range}/n_b$. We compare Δr with σ (calculated based on the n sample values). The number of sample data points falling into each bin k are counted and denoted by S_k (sample number). The n sample values here are the sum of numbers $S_1, \ldots, S_{n_b}$ (i.e. $n = \sum_{k=1}^{n_b} S_k$). The expected number E_k is determined by the assumed distribution of r. For instance if a normal

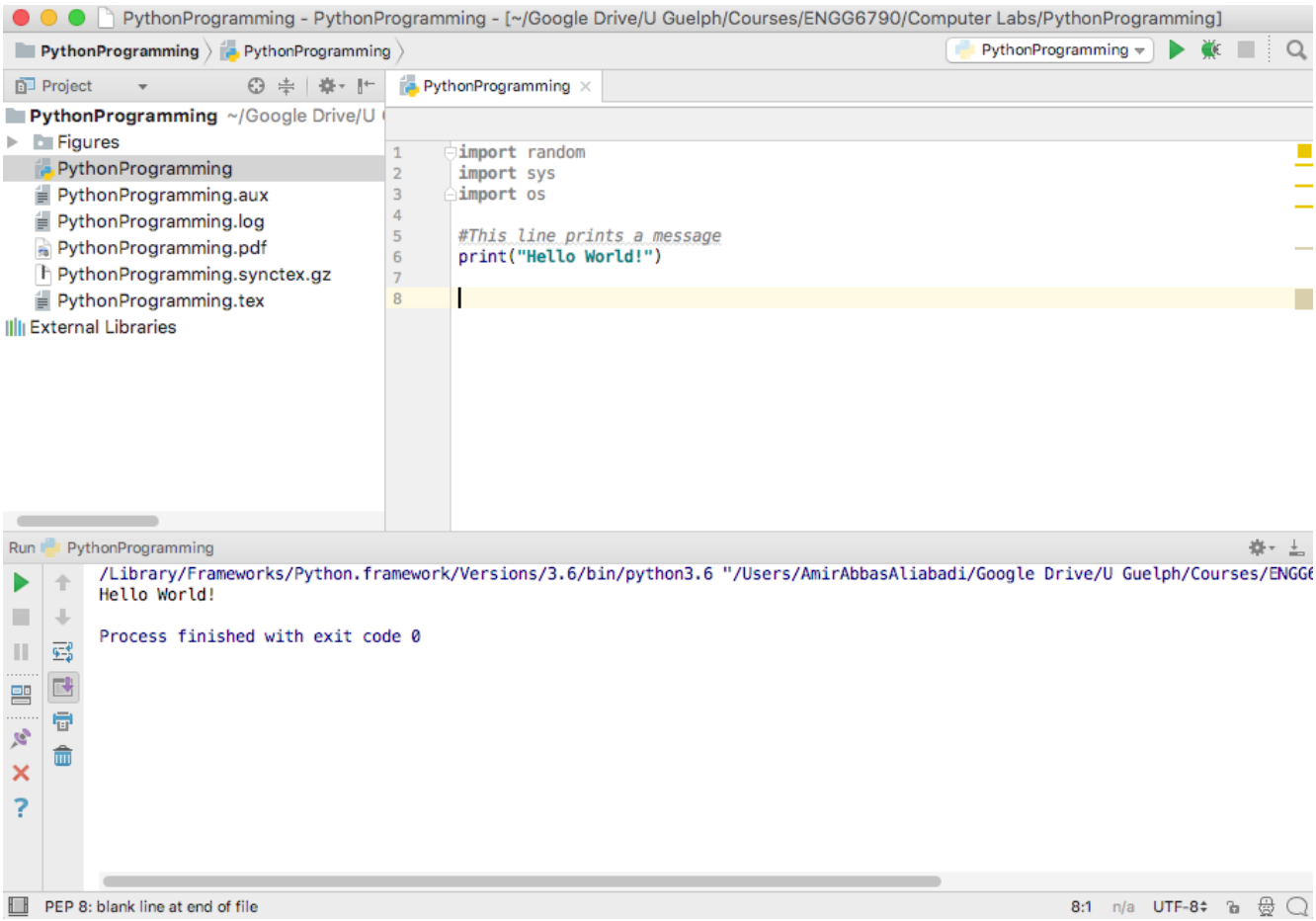

Figure 4: A simple program to print `Hello World!` in the output console.

distribution is assumed, its probability density function can be used to calculate the expected number E_k by integrating this density function over each bin k. The standard deviation is in the order of $\sqrt{E_k}$, therefore, the Chi-squared can be computed by [Aliabadi, 2022]

$$\chi^2 = \sum_{k=1}^{n_b} \frac{(S_k - E_k)^2}{E_k}. \tag{2}$$

If the assumed distribution of r is correct, then χ^2 should be in the order of n_b or less. If $\chi^2 \gg n_b$, then the assumed distribution is probably incorrect. To find the level of agreement between the observed values and expected distribution, we first compute the number of degrees of freedom as [Aliabadi, 2022]

$$\nu = n_b - c\,, \tag{3}$$

where n_b is the number of bins and c is the number of constraints, i.e. location and scale parameters of the distribution, and expected value. For this type of problem $c = 3$, so $\nu = n_b - 3$. Once χ^2 and ν are determined, we then use Table 1 to find the level of confidence P. In this table the first

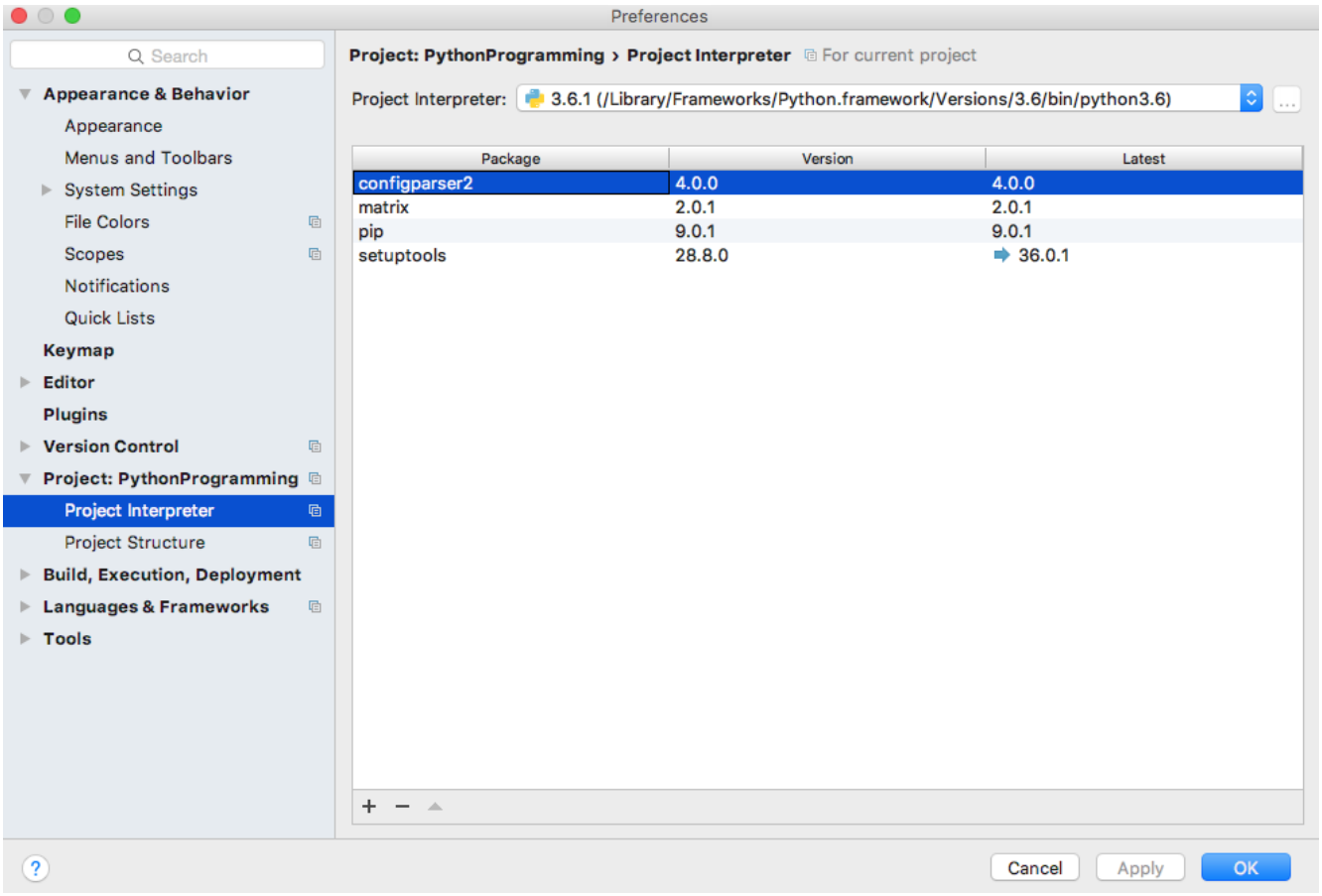

Figure 5: `Preferences` window and the `Project Interpreter`.

column is degrees of freedom ν the first row provides confidence levels P, and the other rows and columns are values of χ^2 [Aliabadi, 2022].

The number of bins n_b must be chosen subject to the constraint that $E_k > 5$, i.e. in each bin at least 5 expected values are considered, corresponding to the 5 sampled values. For $n \geq 25$, and for equal-width bins, the numbers of bins may be estimated using the *Scott's formula* such that

$$n_b = 1.15n^{1/3}, \tag{4}$$

where n is the number of sampled values [Aliabadi, 2022].

In this assignment, we will demonstrate the application of the χ^2 test for monthly returns of the S&P 500 index fund since 1900 until 2022.

6 Python Script

The data set provided in file `SP500Returns.txt` contains 1470 sample points, which provide monthly returns of the S&P 500 index from 1900 until 2022.

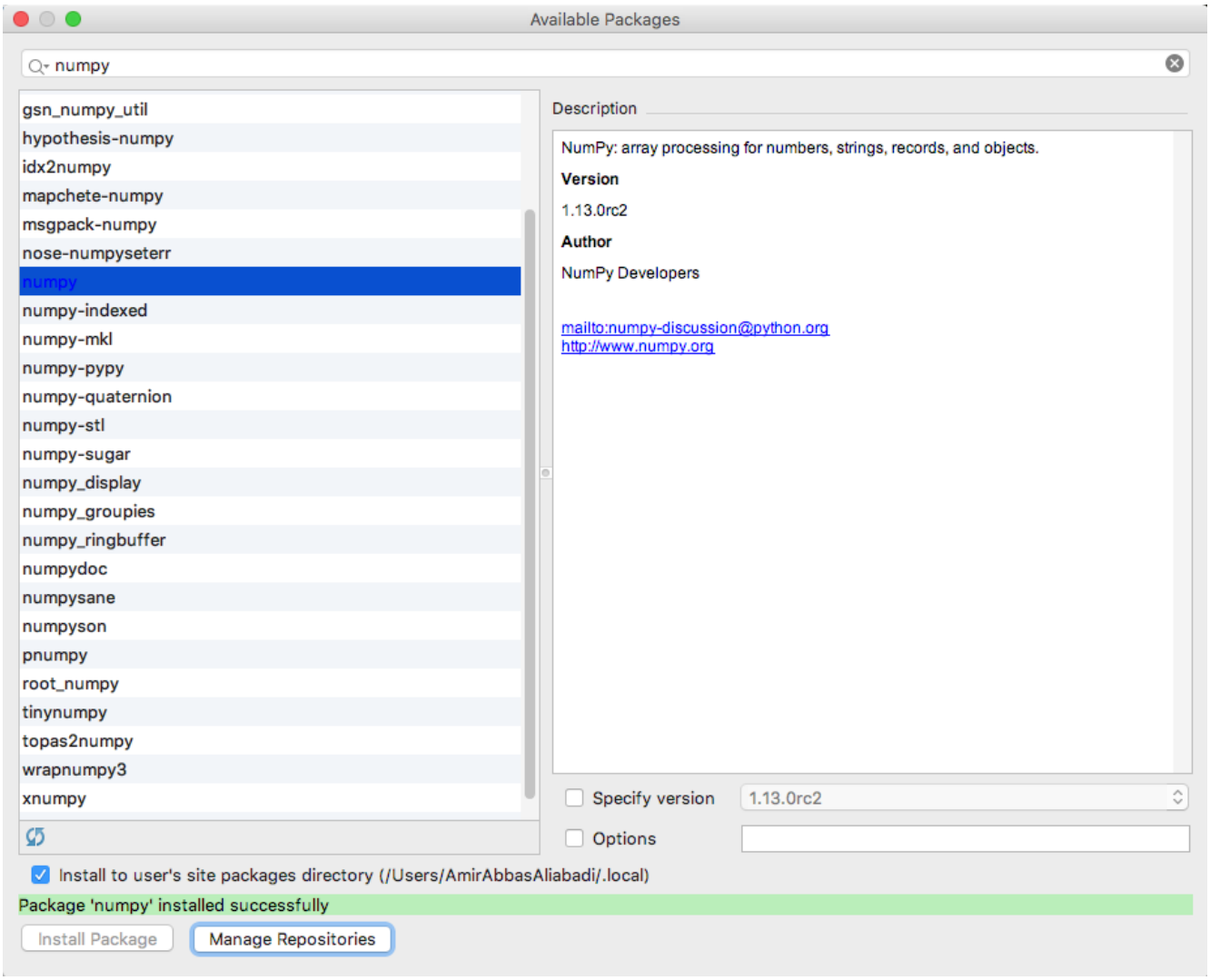

Figure 6: The `numpy` interpreter.

Create a new folder for your project titled `Monthly Returns`. Copy and paste the following code in the `IDE` environment. Name this code `Analysis1.py`. Note that you must have installed the `numpy`, `matplotlib`, and `scipy` packages for this code to work.

```
# Analyze monthly returns of S&P 500
# Chi Squared test for normal and student's t distributions

import numpy
import matplotlib.pyplot as plt
import scipy.stats

inputFileName = "SP500Returns.txt"

# Consider Bins of 5% width from -30% to +30%
BinWidth = 5
```

Table 1: The percentage probability P that the sampled values agree with the expected assumed distribution according to the value of χ^2 with degrees of freedom ν

ν	0.995	0.990	0.975	0.950	0.900	0.750	0.500	0.250	0.100	0.050
1	0.00	0.00	0.00	0.00	0.01	0.10	0.45	1.32	2.71	3.84
2	0.01	0.02	0.05	0.10	0.21	0.57	1.39	2.77	4.61	5.99
3	0.07	0.11	0.21	0.35	0.58	1.21	2.37	4.11	6.25	7.81
4	0.20	0.29	0.48	0.71	1.06	1.92	3.36	5.39	7.78	9.49
5	0.41	0.55	0.83	1.15	1.61	2.67	4.35	6.63	9.24	11.1
6	0.67	0.87	1.24	1.64	2.20	3.45	5.35	7.84	10.6	12.6
7	0.98	1.24	1.69	2.17	2.83	4.25	6.35	9.04	12.0	14.1
8	1.34	1.65	2.18	2.73	3.49	5.07	7.34	10.2	13.4	15.5
9	1.73	2.09	2.70	3.33	4.17	5.90	8.34	11.4	14.7	16.9
10	2.16	2.56	3.25	3.94	4.87	6.74	9.34	12.5	16.0	18.3
11	2.60	3.05	3.82	4.57	5.58	7.58	10.3	13.7	17.3	19.7
12	3.07	3.57	4.40	5.23	6.30	8.44	11.3	14.8	18.5	21.0
13	3.57	4.11	5.01	5.89	7.04	9.30	12.3	16.0	19.8	22.4
14	4.07	4.66	5.63	6.57	7.79	10.2	13.3	17.1	21.1	23.7
15	4.60	5.23	6.26	7.26	8.55	11.0	14.3	18.2	22.3	25.0
16	5.14	5.81	6.91	7.96	9.31	11.9	15.3	19.4	23.5	26.3
17	5.70	6.41	7.56	8.67	10.1	12.8	16.3	20.5	24.8	27.6
18	6.26	7.01	8.23	9.39	10.9	13.7	17.3	21.6	26.0	28.9
19	6.84	7.63	8.91	10.1	11.7	14.6	18.3	22.7	27.2	30.1
20	7.43	8.26	9.59	10.9	12.4	15.5	19.3	23.8	28.4	31.4
30	13.8	15.0	16.8	18.5	20.6	24.5	29.3	34.8	40.3	43.8
40	20.7	22.2	24.4	26.5	29.1	33.7	39.3	45.6	51.8	55.8

```
MinReturn = -30
MaxReturn = +30
nb = int((MaxReturn-MinReturn)/BinWidth)

# Calculate degrees of freedom for student's t and Chi squared distributions
nuStudentt = nb - 3
nuChi2 = nb - 3

# Make a vector for plotting of probability distribution functions
r = numpy.arange(MinReturn, MaxReturn, 0.1)

#Load data into vectors
data = numpy.loadtxt(inputFileName)
nDataset = len(data)

Year = data[:, 0]
Month = data[:, 1]
Return = data[:, 2]

AverageReturn = numpy.mean(Return)
StdReturn = numpy.std(Return)

print('Sample Size: ', nDataset)
print('Average Return [Percent]: ', AverageReturn)
print('Std of Return [Percent]: ', StdReturn)

# Define vector to store the number of samples in each bin
```

```
Sk = numpy.zeros((nb, 1))
Chi2Normal = 0
Chi2Studentt = 0

# Loop through the dataset to populate values of Sk
for k in range(0, nb):
    BinMin = MinReturn + k * BinWidth
    BinMax = MinReturn + (k + 1) * BinWidth

    for j in range(0, nDataset):
        if (Return[j] > BinMin) and (Return[j] <= BinMax):
            Sk[k] = Sk[k] + 1

    # Print bin number, bin min, bin max, and number of samples in bin
    # Must make sure no bin has zero sample count
    print('k, BinMin, BinMax, Sk: ', k, BinMin, BinMax, Sk[k])

    # For normal distribution the expected value can be computed by
    NormalCDFBinMax = scipy.stats.norm.cdf(BinMax,AverageReturn,StdReturn)
    NormalCDFBinMin = scipy.stats.norm.cdf(BinMin,AverageReturn,StdReturn)
    ExpectedNormal = nDataset * (NormalCDFBinMax - NormalCDFBinMin)
    Chi2Normal = Chi2Normal + ((Sk[k]-ExpectedNormal)**2) / ExpectedNormal

    # For student's t distribution the expected value can be computed by
    StudenttCDFBinMax = scipy.stats.t.cdf(BinMax, nuStudentt, AverageReturn,
        StdReturn)
    StudenttCDFBinMin = scipy.stats.t.cdf(BinMin, nuStudentt, AverageReturn,
        StdReturn)
    ExpectedStudentt = nDataset * (StudenttCDFBinMax - StudenttCDFBinMin)
    Chi2Studentt = Chi2Studentt + \
        ((Sk[k] - ExpectedStudentt) ** 2) / ExpectedStudentt

print('Student\'s t Distribution Degrees of Freedom: ', nuStudentt)
print('Chi Squared Distribution Degrees of Freedom: ', nuChi2)
print('Chi Squared Assuming Normal Distribution: ', Chi2Normal)
print('Chi Squared Assuming Student\'s t Distribution: ', Chi2Studentt)

fig = plt.figure(figsize = (10,6))
plt.rc('xtick', labelsize = 16)
plt.rc('ytick', labelsize = 16)
plt.hist(Return, density = True, color = 'cyan', edgecolor = 'black', bins = 100,
    label='Actual Return')
plt.plot(r, scipy.stats.norm.pdf(r, AverageReturn, StdReturn), color = 'red',
    linewidth = 3, label='Normal Distribution')
plt.plot(r, scipy.stats.t.pdf(r, nuStudentt, AverageReturn, StdReturn),
    color = 'blue', linewidth = 3, label='Student\'s t Distribution')
```

```
plt.legend(loc='upper right', fontsize=20)
plt.xlabel('Return [Percent]', fontsize = 20)
plt.ylabel('Probability [-]',fontsize = 20)
plt.tick_params(axis = 'both', which = 'major', labelsize = 16)
plt.tight_layout()
plt.savefig('SP500Returns.pdf', dpi = 600, bbox_inches = 'tight')
fig.show()

plt.show()
```

You must ensure that the file `SP500Returns.txt` is located in the same directory as `Analysis1.py`. Upon running this code, you should obtain the following output in the console:

```
Sample Size:  1470
Average Return [Percent]:  0.8680272108843538
Std of Return [Percent]:  4.253252671188704
k, BinMin, BinMax, Sk:  0 -30 -25 [1.]
k, BinMin, BinMax, Sk:  1 -25 -20 [2.]
k, BinMin, BinMax, Sk:  2 -20 -15 [2.]
k, BinMin, BinMax, Sk:  3 -15 -10 [23.]
k, BinMin, BinMax, Sk:  4 -10 -5 [63.]
k, BinMin, BinMax, Sk:  5 -5 0 [439.]
k, BinMin, BinMax, Sk:  6 0 5 [801.]
k, BinMin, BinMax, Sk:  7 5 10 [125.]
k, BinMin, BinMax, Sk:  8 10 15 [10.]
k, BinMin, BinMax, Sk:  9 15 20 [1.]
k, BinMin, BinMax, Sk:  10 20 25 [1.]
k, BinMin, BinMax, Sk:  11 25 30 [1.]
Student's t Distribution Degrees of Freedom:  9
Chi Squared Distribution Degrees of Freedom:  9
Chi Squared Assuming Normal Distribution:  [1249931.83937987]
Chi Squared Assuming Student's t Distribution:  [191.49851207]

Process finished with exit code 0
```

The sample average and standard deviation are $R = 0.87\%$ and $\sqrt{V} = 4.25\%$, respectively. We have defined $n_b = 12$ bins (from 0 to 11) with a bin width of $\Delta r = 5\%$ from -30% to $+30\%$. This ensures that at least one sample will fall in each bin although most of the samples fall in the central bins. Note that a few data points beyond this range of bins are excluded from the analysis. We can ensure that $\Delta r \sim \sqrt{V}$.

For application of the normal distribution, we use the same sample average and standard deviation as the location and scale parameter. The probabilities of expected values can be computed using the `scipy.stats.norm.cdf()` function. This function takes the the `BinMin` or `BinMax` values, `AverageReturn`, and `StdReturn` to give the cumulative probability of random variable being from $-\infty$ to `BinMin` or `BinMax`. Note that to find the probability of the random variable being between

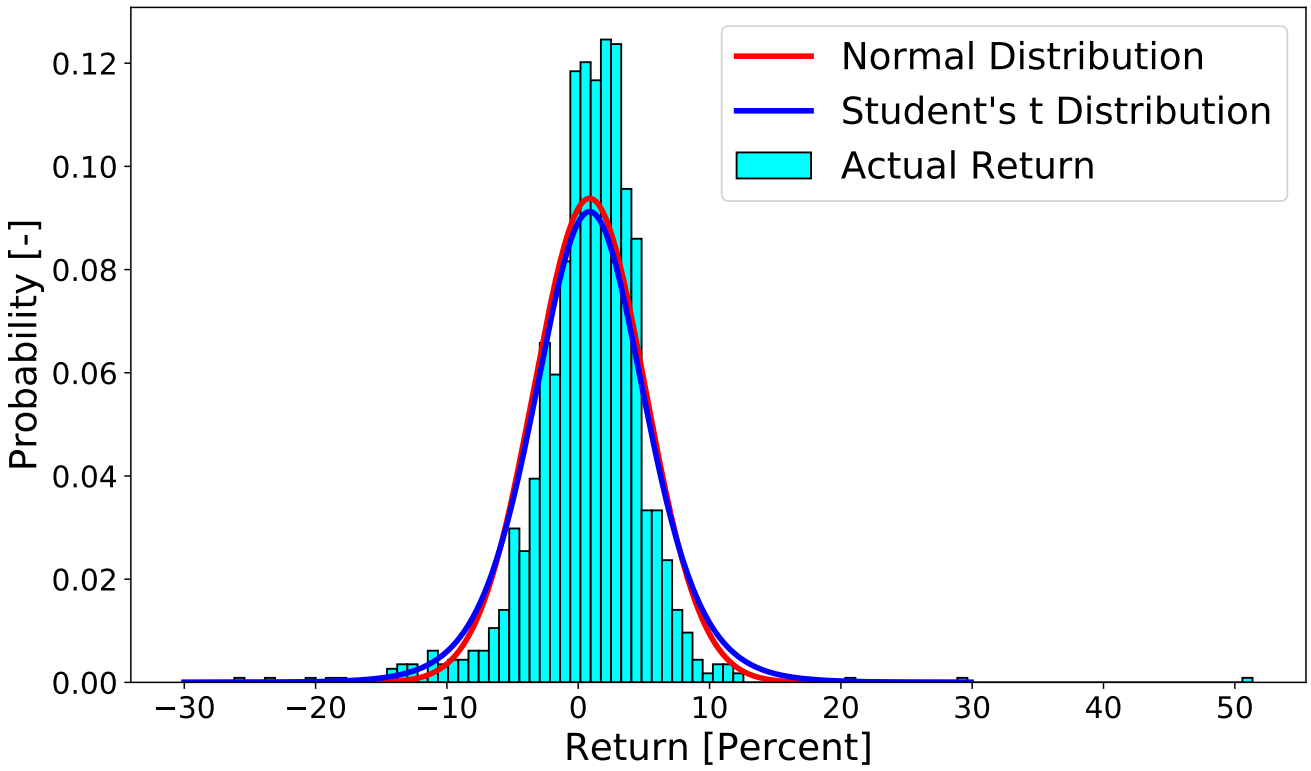

Figure 7: Historical monthly returns, in percent, of the S&P 500 index from 1900 until 2022; normal probability distribution function with the same average and standard deviation; student's t distribution function with the same average, standard deviation, and degrees of freedom of 9

`BinMin` and `BinMax` we must find the difference in the respective cumulative probabilities.

For application of the student's t distribution, we use the same location and scale parameters. In addition, we assume $\nu = 9$ for the degrees of freedom, and the probabilities of expected values can be computed using `scipy.stats.t.cdf()` function. In addition to the variables noted for the nurmal distribution, this function also takes the degrees of freedom `nuStudentt`.

The χ^2 is then computed for each case. The code also draws the `SP500Returns.pdf` figure and saves it in the same directory. Figure 7 shows the probability density of the actual returns in percent.

As can be seen, the actual data set has very wide tails, with returns as low as -30% and as high as $+50\%$. The figure also shows the normal and student's t probability functions. Note that neither function fits the central peak, nor do they fit the wide tails; however, at least the student's t distribution features wider tails. For application of the χ^2 test, we have $\nu = 9$. If we compute the value of χ^2 we find $\chi^2 = 1249932$ and $\chi^2 = 191$ for the normal and student's t distribution functions, respectively. Consulting Table 1 we find that neither probability distribution function fits the data well. In fact the probability of agreement P is far less than even 0.05. However, at least the student's t distribution results in a much lower χ^2, and it is a much better choice than the normal distribution. Note that many other combinations of return ranges and bin widths can

be attempted for this test, but this simple demonstration shows the far better applicability of the student's t distribution than the normal distribution. At least a financial analyst can hope that during shorter time periods of analysis no exuberant or catastrophic market events may occur, so that the student's t distribution is a practical model.

7 Task Activity

Now that you can successfully run the python code and generate the results. Perform the following analyses by manipulating the python code `Analysis1.py` and/or the text file `SP500Returns.txt`.

Produce the statistics of the monthly return, i.e. average return and standard deviation of return, for each decade: 1900-1909, 1910-1919, ..., 2010-2019. Produce the statistics of the monthly return, i.e. average return and standard deviation of return, for periods of three decades: 1900-1929, 1930-1959, 1960-1989, 1990-2019. List the statistics in a table.

Apply the χ^2 test for each period discussed above using both the normal distribution and student's t distribution. Report the χ^2 values for each case in a table. In addition, provide figures for each case, showing the probability distribution for the data, the normal distribution, and the student's t distribution.

Try to answer the following questions:

1. Is there a relationship between the length of the data set, i.e. one decade, three decades, or the entire data set, and the magnitude of the statistics of the monthly return, i.e. average return and standard deviation of return?

2. Is there a relationship between the length of the data set, i.e. one decade, three decades, or the entire data set, and the magnitude of the χ^2?

3. For each case, is the normal distribution or the student's t distribution describe the probability of the monthly return better?

4. For the χ^2 test of the student's t distribution, how did you determine the degrees of freedom?

5. How did you determine the width of the bins in each case?

6. Is the χ^2 test going to work in case there are no sample data points in a particular bin?

Hint: to decide for the number of bins, you can use the Scott's formula. Certainly, the bin width could be wider for each case because the number of data points in each decade or a three-decade period will be fewer than the number of data points for the entire data set.

Hint: for each analysis it is advisable not to overwrite the original files, but to simply create new ones, e.g. `Analysis2.py`, `Analysis2.py`, ..., `SP500Returns1900-1909.txt`, ...

8 Reporting Guidelines

Please provide your report as a single PDF file. Do not submit your python code or data set as separate attachments. Instead, list your python code in the body of your report in the Appendix. The Microsoft Word format will not be accepted. Your report may include the following sections: Introduction, Methodology, Results and Discussions, Conclusion, and Appendix.

I *strongly discourages* use of Microsoft Excel for technical communications. You should perform any data analysis or plotting using Python. If you use Microsoft Excel in any part of your analysis or reporting, your report will not be graded.

Students are *strongly encouraged* if they use LaTeX(instead of Microsoft Word) to generate the report. I will give two bonus marks for using LaTeX. I generally suggest the open source Text Studio for production of reports, manuscripts, and theses in LaTeX. The program can be downloaded via `https://www.texstudio.org/`.

References

[Aliabadi, 2022] Aliabadi, A. A. (2022). *Turbulence: A Fundamental Approach for Scientists and Engineers*. Springer, Cham.

ENGG*6405: Finance and Economics for Engineers

Portfolio Selection

Amir A. Aliabadi

This document is typeset using LaTeX

April 9, 2025

1 Introduction

Portfolio theory attempts to identify different types of risks and returns that are associated with a group of investments. Once the relevant risks and returns are identified, a risk management strategy is used to 1) retain a risk, 2) neutralize a risk, or 3) transfer a risk [Fabozzi and Jones, 2019]. This is achieved by the proper design of the portfolio, so the desired levels of risk and return are obtained. Generally it is desired to lower the risk and increase the return, but there are always trade-offs. The American economist *Harry Markowitz* pioneered the modern portfolio theory in his seminal paper [Markowitz, 1952], who was subsequently awarded the Nobel prize in economics.

One way to quantify total risk is to consider the past variations of the share price of a company. It must be cautioned that this approach has limited utility because past movements of the share price does not indicate what the future movements will be; however, for lack of other options we can quantify the risk given past share price movements. The pioneering work in this concept was performed by [Markowitz, 1952, Markowitz, 1959], who defined risk as the variance V of share price movements in an arbitrary time period. For instance, the risk, or variance V, can be calculated over a day, week, month, year, or decade.

By the same token, the return of an investment may be studied using historical share price movements, again, with the caveat that past movements do not indicated attributes of future movements. Price movements can be considered for any given time period T. For instance, the time period can be a month, for which we can calculate the monthly *average share return* R_s. Of course, this return excludes any *average dividend return* paid by the company to the investor over that time period, but an average dividend per share and per the same time period D can be considered. The *average total return* will be

$$R = R_s + D. \tag{1}$$

The return can be computed by considering buying and selling of shares at the beginning and end of a time period. A sliding window can be defined where the selected time period can move

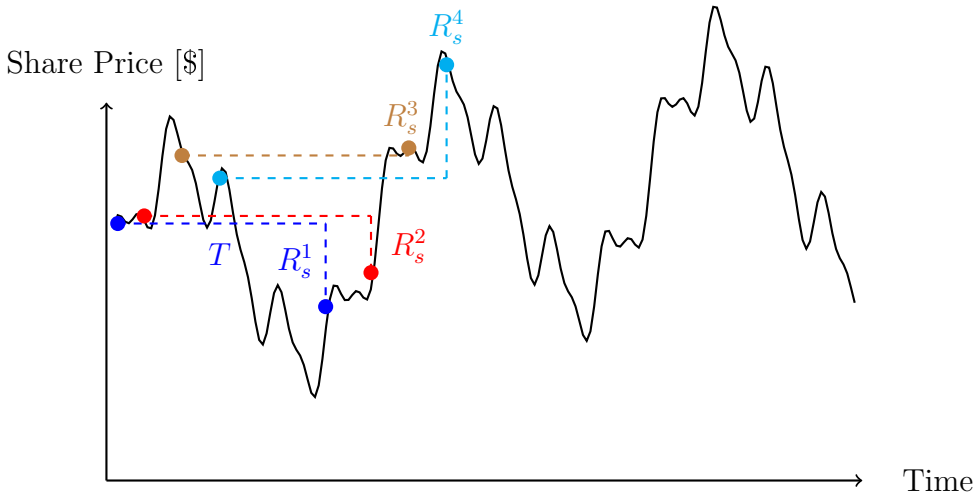

Figure 1: Visualization showing successive share returns R_s^i for investments in successive but inclusive time periods T

to the right continuously. This way, a larger statistical sample can be collected to compute the average share return R_s. This approach is shown in Figure 1. As seen in the figure, there could be significant overlap among successive time periods. The average share return can be taken to give the total return, after adding the dividend per time period, such that

$$R = \frac{\sum_{i=1}^{n} R_s^i}{n} + D = R_s + D, \tag{2}$$

where n is the total number of time periods in the sample. If we divide each share return by the share price at the beginning of each time period, and if we express the dividend normalized by the share price, then the computed total return would be normalized.

Portfolio allocation refers to splitting a total amount among various investments with unequal proportions. For instance, a total amount of \$1,000 may be invested in two companies 1 and 2. Suppose that the first investment is \$300 in company 1, and the second investment is \$700 in company 2. The fraction of overall investment for companies 1 and 2 are $X_1 = 0.3$ and $X_2 = 0.7$. Note that $X_1 + X_2 = 1$. In a general sense, a total amount I can be invested in N companies with fraction $X_i \geq 0$ associated with company i. The total investment amount I and the condition on X_i can be written as

$$I = \sum_{i=1}^{N} I_i = \sum_{i=1}^{N} X_i I = X_1 I + X_2 I + \cdots + X_N I, \tag{3}$$

$$\sum_{i=1}^{N} X_i = X_1 + X_2 + \cdots + X_N = 1. \tag{4}$$

In this way a portfolio is allocated among N investments. There are some key questions to ask

when deciding to determine the values of X_i. For example, what is the overall return R of the portfolio given historical records? what is the overall risk, or variance V, of the portfolio given historical records? These questions will be addressed in the next section.

[Markowitz, 1952] shows that, for given values of $X_i \geq 0$, the overall return of a portfolio will be the weighted average of individual returns, i.e.

$$R = \sum_{i=1}^{N} X_i R_i = X_1 R_1 + X_2 R_2 + \cdots + X_N R_N, \tag{5}$$

where R_i is the share plus dividend return of investment i over the time period for historical analysis. Computation of the risk, or variance V, for the portfolio requires more work. This requires pairwise analysis of every two companies in the investment. The covariance between company i and j can be given by

$$C_{ij} = \frac{\sum_{k=1}^{n}(R_i^k - R_i)(R_j^k - R_j)}{n}, \tag{6}$$

where R_i and R_j are the average share plus dividend return of companies i and j over the historical record, while R_i^k and R_j^k are specific returns during time period k. This covariance is still calculated by an average over n records. In a way, the covariance is a form of average for the product of deviations of two sample variables from their respective averages. Note that if $i = j$, the covariance in Equation 6 actually gives us the variance. [Markowitz, 1952] shows that, for given values of $X_i \geq 0$, the overall risk, or variance V, of a portfolio is given by

$$V = \sum_{i=1}^{N}\sum_{j=1}^{N} X_i X_j C_{ij} = X_1 X_1 C_{11} + X_1 X_2 C_{12} + \cdots + X_N X_N C_{NN}. \tag{7}$$

This summation is over all combinations of i and j. Generally, the larger C_{ij} is, then the combination of investments in companies i and j will *increase* the risk, or variance V, of the portfolio; while the smaller C_{ij} is (it could also be negative), then the combination of investments in companies i and j will *decrease* the risk, or variance V, of the portfolio.

Any sensible investor is interested to know the answer to the following two questions, when designing a portfolio, i.e. determining the weights X_i:

1. For a given value of portfolio risk, or variance V, what are the values of weights X_i that give the *highest* portfolio return R?

2. For a given value of portfolio return R, what are the values of weights X_i that give the *lowest* portfolio risk, or variance V?

These questions lead to mathematical optimization problems that are solved by the *method of Lagrange multipliers*, which is a strategy for finding the local maximum or minimum of a function subject to equality constraints (i.e., subject to the condition that one or many equations have to

be satisfied exactly by the chosen values of the variables.). In this case the variables required for optimization are X_i. Let us frame the portfolio questions above mathematically

1. Maximize R by finding variables X_i that meet the condition $V = V_*$.

2. Minimize V by finding variables X_i that meet the condition $R = R_*$.

Here V_* is an attainable and realistic portfolio risk that the investor is comfortable with, and R_* is an attainable and realistic portfolio return that the investor is comfortable with. Rather than the method of Lagrange multipliers, a stochastic numerical way can be invented to find an approximate numerical solution to the optimization problem.

Suppose that $N = 10$, so we have to determine X_1, X_2, through X_{10}. We can generate thousands, perhaps millions, of portfolios with random values of X_i. To do so, we define a quantized amount of weight to be $\Delta X = 0.1$. This means that the weights would be randomly assigned to X_1, X_2, through X_{10}, by adding $\Delta X = 0.1$ to any of the X_i values starting from zero. This random addition repeats 10 times, so by the end $\sum_{i=1}^{N} X_i = 1$. For example, to generate the first random portfolio, and before adding any quantized amount of weight, the portfolio is specified by

$$X_1 = 0.0, X_2 = 0.0, X_3 = 0.0, X_4 = 0.0, X_5 = 0.0,$$
$$X_6 = 0.0, X_7 = 0.0, X_8 = 0.0, X_9 = 0.0, X_{10} = 0.0.$$

Suppose that the first ΔX addition is randomly assigned to X_3. Then we will have

$$X_1 = 0.0, X_2 = 0.0, X_3 = 0.1, X_4 = 0.0, X_5 = 0.0,$$
$$X_6 = 0.0, X_7 = 0.0, X_8 = 0.0, X_9 = 0.0, X_{10} = 0.0.$$

Suppose that the second ΔX addition is randomly assigned to X_5. Then we will have

$$X_1 = 0.0, X_2 = 0.0, X_3 = 0.1, X_4 = 0.0, X_5 = 0.1,$$
$$X_6 = 0.0, X_7 = 0.0, X_8 = 0.0, X_9 = 0.0, X_{10} = 0.0.$$

The process continues until the quantized weight addition is repeated 10 times. In the end some weights may experience multiple additions, while some other weights may experience no additions. For instance, by the end, the first portfolio sample may result in

$$X_1 = 0.0, X_2 = 0.6, X_3 = 0.1, X_4 = 0.0, X_5 = 0.1,$$
$$X_6 = 0.2, X_7 = 0.0, X_8 = 0.0, X_9 = 0.0, X_{10} = 0.0.$$

After generating a large set of portfolios, we are destined to have created portfolios with multiple weight preferences. This techniques will not generate *all* the portfolio possibilities with $\Delta X = 1$, but it is good enough to give us *many* various combinations of weights that may be possible.

After generating a large set of portfolios, we are destined to have created portfolios with multiple weight preferences. This techniques will not generate *all* the portfolio possibilities with $\Delta X = 1$, but it is good enough to give us *many* various combinations of weights that may be possible.

If we plot a map for all the generated portfolios, then we can observe the risk versus return behavior of each portfolio and decide which one suits our investment preferences given the questions asked above.

In this assignment, we will demonstrate the application of the portfolio theory. Suppose we consider the following $N = 10$ investments with share tickers (for investment 1 to 10) of ZSP.TO, IVOO, VIOO, ZCN.TO, VEE.TO, VIU.TO, ZAG.TO, VBG.TO, VBU.TO, and HEP.TO. These are Exchange Traded Funds (ETFs). The first three tickers (ZSP.TO, IVOO, VIOO) are investments in the United States according to S&P indexes for 500 LargeCap, 400 MidCap, and 600 SmallCap companies, respectively. The following three tickers (ZCN.TO, VEE.TO, VIU.TO) are investments in Canada, emerging markets, and developed markets, respectively, in respective companies. The next three tickers (ZAG.TO, VBG.TO, VBU.TO) are aggregate bonds in Canada, other international countries, and the United States, respectively. The final ticker (HEP.TO) is investment in gold. We wish to create the correlation matrix for these shares as well as the risk versus return map.

2 Python Script

One way to obtain share price data is using the Yahoo Finance website with the address `https://ca.finance.yahoo.com/`. In the search box, you can type `ZSP.TO`, as an example for the first investment. Click on `Historical Data`, adjust the `Time Period` to the desirable range, and click `Apply`. This will show the share prices. Click `Download`, and your can obtain the share prices in the spreadsheet format. The column that we desire is the `close` price. We can assemble a history of share prices, such as provided in the file `StockHistory-10-2022-05-05.xls`, which will contain the share prices for the 10 investments noted above for a period of 6 years, ending as of May 5, 2022. This data should be saved as `Text (Tab delimited) (*.txt)`, which is provided in file `StockHistory-10-2022-05-05.txt`. Note that the first line in the data set indicates the share tickers:

```
#ZSP IVOO VIOO ZCN VEE VIU ZAG VBG VBU HEP
```

To obtain the average annual dividends for these investments, one can use various sources of data. For instance, according to the iPhone's `Stocks` application the following dividends can be obtained.

```
StockDividend = [0.0130, 0.0107, 0.0088, 0.0281, 0.0165, 0.0228, 0.0303,
0.0067, 0.0204, 0.0682]
```

Create a new folder for your project titled `Portfolio Selection`. Copy and paste the following code in the `IDE` environment. Name this code `Analysis1.py`. Note that you must have installed the `numpy` and `random` packages for this code to work.

```
#Compute asset allocation and the resulting risk versus return based on
#Markowitz 1952
#Take closing price of 10 stocks
#This code uses a randomized approach to assign the portfolio weights

#Import libraries
import numpy
import random

#Analysis 2022-05-05
inputFileName = "StockHistory-10-2022-05-05.txt"
outputFileName = "PortfolioAnalysis-10-2022-05-05.txt"

#Symbols:
#ZSP.TO, IVOO, VIOO, ZCN.TO, VEE.TO, VIU.TO, ZAG.TO, VBG.TO, VBU.TO, HEP.TO
StockIndex = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
StockDividend = [0.0130, 0.0107, 0.0088, 0.0281, 0.0165, 0.0228, 0.0303,
    0.0067, 0.0204, 0.0682]

#Load data into vectors
data = numpy.loadtxt(inputFileName)
nDataset = len(data)

nYears = 6
nStocks = 10
nSample = 500000
# For monthly return analysis consider this value for number of days in a month
returnFrequencyDays = int(nDataset / (nYears * 12))
daysInYear = int(nDataset/nYears)

#Define price matrix to be filled later
StockPrice = numpy.zeros((nDataset, nStocks))
for i in range(0, nStocks):
    for j in range(0, nDataset):
        StockPrice[j][i] = data[j, StockIndex[i]]

#First calculate the stock price return, i.e. share gain, through price history
StockShareGain = numpy.zeros((nDataset-returnFrequencyDays, nStocks))
StockReturn = numpy.zeros((nDataset-returnFrequencyDays, nStocks))

x = [i for i in range(0, returnFrequencyDays)]
```

```
for i in range(0, nStocks):
    #Find the returns in each of the time periods
    for j in range(0, nDataset-returnFrequencyDays):
        StockShareGain[j][i] = (StockPrice[j + returnFrequencyDays, i] -
            StockPrice[j, i]) / StockPrice[j][i]
        StockReturn[j][i] = StockShareGain[j][i] + \
            StockDividend[i]*returnFrequencyDays/daysInYear

MeanStockReturn = numpy.zeros((nStocks))
CovStockReturn = numpy.zeros((nStocks, nStocks))

# Set vectors y and z
Vectory = numpy.zeros((nDataset-returnFrequencyDays))
Vectorz = numpy.zeros((nDataset-returnFrequencyDays))

#print the return of the selected stocks
for i in range(0, nStocks):
    MeanStockReturn[i] = numpy.mean(StockReturn[:, i])
    print('Stock Index, Average Monthly Return [Percent]: ',
        i, numpy.round(100 * MeanStockReturn[i], 2))

for y in range(0, nStocks):
    for z in range(0, nStocks):
        for x in range(0, nDataset-returnFrequencyDays):
            Vectory[x] = StockReturn[x][y]
            Vectorz[x] = StockReturn[x][z]
        Covariance = numpy.cov(Vectory, Vectorz)
        CovStockReturn[y][z] = Covariance[0][1]
        print('Stock Index y, z, Covariance of Return [Percent Squared]: ',
            y, z, numpy.round(10000 * CovStockReturn[y][z], 1))

#Now we should sample random portfolios for
#weights X0, X1, ... , X9 >= 0 subject to sum Xi = 1
X = numpy.zeros((nSample, nStocks))
Return = numpy.zeros((nSample))
Risk = numpy.zeros((nSample))

#Find random combinations of X0, X2, ..., X9
#assuming Delta X = 0.1, subject to Sum Xi = 1

#Loop over sample
for a in range(0, nSample):
    #Random selection of stock
    for b in range(0, nStocks):
        RandStock = random.randint(0, nStocks-1)
        X[a][RandStock] = X[a][RandStock] + 1/nStocks
```

```
    #Calculate the sum of X0 to X9 and warn user if not close to 1
    SumX = numpy.sum(X[a][:])
    if SumX > 1.001 or SumX < 0.999:
        print('Warning, Sum of X[a][:] is not close to 1, it is: ', SumX)

    Return[a] = X[a][0] * MeanStockReturn[0] + \
        X[a][1] * MeanStockReturn[1] + \
        X[a][2] * MeanStockReturn[2] + \
        X[a][3] * MeanStockReturn[3] + \
        X[a][4] * MeanStockReturn[4] + \
        X[a][5] * MeanStockReturn[5] + \
        X[a][6] * MeanStockReturn[6] + \
        X[a][7] * MeanStockReturn[7] + \
        X[a][8] * MeanStockReturn[8] + \
        X[a][9] * MeanStockReturn[9]

    # To calculate risk, all the variances and covariances must be considered
    for y in range(0, nStocks):
        for z in range(0, nStocks):
            Risk[a] = Risk[a] + X[a][y] * X[a][z] * CovStockReturn[y][z]

#Write data to file
outputFile = open(outputFileName, "w")
outputFile.write("#Sample \t Risk \t Return \t "
    "X0 \t X1 \t X2 \t X3 \t X4 \t "
    "X5 \t X6 \t X7 \t X8 \t X9 \n")

#Now loop through data points and write to file
for a in range(0, nSample):
    outputFile.write("%i \t %f \t %f \t "
        "%f \t %f \t %f \t %f \t %f \t "
        "%f \t %f \t %f \t %f \t %f \n"
    % (a, Risk[a], Return[a],
        X[a][0], X[a][1], X[a][2], X[a][3], X[a][4],
        X[a][5], X[a][6], X[a][7], X[a][8], X[a][9]))

#Close file
outputFile.close()
```

After running this script, the following console output should be generated. Note that the script also generates the file `PortfolioAnalysis-10-2022-05-05.txt`, which contains the return, risk, and investment weightings for 500,000 sample portfolios.

```
Stock Index, Average Monthly Return [Percent]:  0 1.14
Stock Index, Average Monthly Return [Percent]:  1 1.07
```

```
Stock Index, Average Monthly Return [Percent]:  2 1.16
Stock Index, Average Monthly Return [Percent]:  3 0.9
Stock Index, Average Monthly Return [Percent]:  4 0.58
Stock Index, Average Monthly Return [Percent]:  5 0.56
Stock Index, Average Monthly Return [Percent]:  6 0.08
Stock Index, Average Monthly Return [Percent]:  7 -0.05
Stock Index, Average Monthly Return [Percent]:  8 0.05
Stock Index, Average Monthly Return [Percent]:  9 0.97
Stock Index y, z, Covariance of Return [Percent Squared]:  0 0 16.2
Stock Index y, z, Covariance of Return [Percent Squared]:  0 1 15.6
Stock Index y, z, Covariance of Return [Percent Squared]:  0 2 16.0
Stock Index y, z, Covariance of Return [Percent Squared]:  0 3 14.2
Stock Index y, z, Covariance of Return [Percent Squared]:  0 4 11.5
Stock Index y, z, Covariance of Return [Percent Squared]:  0 5 13.5
Stock Index y, z, Covariance of Return [Percent Squared]:  0 6 2.7
Stock Index y, z, Covariance of Return [Percent Squared]:  0 7 1.5
Stock Index y, z, Covariance of Return [Percent Squared]:  0 8 1.3
Stock Index y, z, Covariance of Return [Percent Squared]:  0 9 5.8
Stock Index y, z, Covariance of Return [Percent Squared]:  1 0 15.6
Stock Index y, z, Covariance of Return [Percent Squared]:  1 1 33.2
Stock Index y, z, Covariance of Return [Percent Squared]:  1 2 35.7
Stock Index y, z, Covariance of Return [Percent Squared]:  1 3 19.4
Stock Index y, z, Covariance of Return [Percent Squared]:  1 4 13.1
Stock Index y, z, Covariance of Return [Percent Squared]:  1 5 14.6
Stock Index y, z, Covariance of Return [Percent Squared]:  1 6 2.4
Stock Index y, z, Covariance of Return [Percent Squared]:  1 7 1.6
Stock Index y, z, Covariance of Return [Percent Squared]:  1 8 0.8
Stock Index y, z, Covariance of Return [Percent Squared]:  1 9 5.7
Stock Index y, z, Covariance of Return [Percent Squared]:  2 0 16.0
Stock Index y, z, Covariance of Return [Percent Squared]:  2 1 35.7
Stock Index y, z, Covariance of Return [Percent Squared]:  2 2 41.0
Stock Index y, z, Covariance of Return [Percent Squared]:  2 3 20.6
Stock Index y, z, Covariance of Return [Percent Squared]:  2 4 13.2
Stock Index y, z, Covariance of Return [Percent Squared]:  2 5 15.0
Stock Index y, z, Covariance of Return [Percent Squared]:  2 6 1.9
Stock Index y, z, Covariance of Return [Percent Squared]:  2 7 1.5
Stock Index y, z, Covariance of Return [Percent Squared]:  2 8 0.4
Stock Index y, z, Covariance of Return [Percent Squared]:  2 9 1.1
Stock Index y, z, Covariance of Return [Percent Squared]:  3 0 14.2
Stock Index y, z, Covariance of Return [Percent Squared]:  3 1 19.4
Stock Index y, z, Covariance of Return [Percent Squared]:  3 2 20.6
Stock Index y, z, Covariance of Return [Percent Squared]:  3 3 18.8
Stock Index y, z, Covariance of Return [Percent Squared]:  3 4 11.9
Stock Index y, z, Covariance of Return [Percent Squared]:  3 5 13.6
Stock Index y, z, Covariance of Return [Percent Squared]:  3 6 2.7
Stock Index y, z, Covariance of Return [Percent Squared]:  3 7 1.4
```

Stock Index y, z, Covariance of Return [Percent Squared]: 3 8 1.5
Stock Index y, z, Covariance of Return [Percent Squared]: 3 9 12.2
Stock Index y, z, Covariance of Return [Percent Squared]: 4 0 11.5
Stock Index y, z, Covariance of Return [Percent Squared]: 4 1 13.1
Stock Index y, z, Covariance of Return [Percent Squared]: 4 2 13.2
Stock Index y, z, Covariance of Return [Percent Squared]: 4 3 11.9
Stock Index y, z, Covariance of Return [Percent Squared]: 4 4 18.6
Stock Index y, z, Covariance of Return [Percent Squared]: 4 5 13.6
Stock Index y, z, Covariance of Return [Percent Squared]: 4 6 2.5
Stock Index y, z, Covariance of Return [Percent Squared]: 4 7 1.2
Stock Index y, z, Covariance of Return [Percent Squared]: 4 8 1.5
Stock Index y, z, Covariance of Return [Percent Squared]: 4 9 8.8
Stock Index y, z, Covariance of Return [Percent Squared]: 5 0 13.5
Stock Index y, z, Covariance of Return [Percent Squared]: 5 1 14.6
Stock Index y, z, Covariance of Return [Percent Squared]: 5 2 15.0
Stock Index y, z, Covariance of Return [Percent Squared]: 5 3 13.6
Stock Index y, z, Covariance of Return [Percent Squared]: 5 4 13.6
Stock Index y, z, Covariance of Return [Percent Squared]: 5 5 15.8
Stock Index y, z, Covariance of Return [Percent Squared]: 5 6 2.5
Stock Index y, z, Covariance of Return [Percent Squared]: 5 7 1.2
Stock Index y, z, Covariance of Return [Percent Squared]: 5 8 1.3
Stock Index y, z, Covariance of Return [Percent Squared]: 5 9 4.9
Stock Index y, z, Covariance of Return [Percent Squared]: 6 0 2.7
Stock Index y, z, Covariance of Return [Percent Squared]: 6 1 2.4
Stock Index y, z, Covariance of Return [Percent Squared]: 6 2 1.9
Stock Index y, z, Covariance of Return [Percent Squared]: 6 3 2.7
Stock Index y, z, Covariance of Return [Percent Squared]: 6 4 2.5
Stock Index y, z, Covariance of Return [Percent Squared]: 6 5 2.5
Stock Index y, z, Covariance of Return [Percent Squared]: 6 6 2.6
Stock Index y, z, Covariance of Return [Percent Squared]: 6 7 1.4
Stock Index y, z, Covariance of Return [Percent Squared]: 6 8 1.6
Stock Index y, z, Covariance of Return [Percent Squared]: 6 9 5.7
Stock Index y, z, Covariance of Return [Percent Squared]: 7 0 1.5
Stock Index y, z, Covariance of Return [Percent Squared]: 7 1 1.6
Stock Index y, z, Covariance of Return [Percent Squared]: 7 2 1.5
Stock Index y, z, Covariance of Return [Percent Squared]: 7 3 1.4
Stock Index y, z, Covariance of Return [Percent Squared]: 7 4 1.2
Stock Index y, z, Covariance of Return [Percent Squared]: 7 5 1.2
Stock Index y, z, Covariance of Return [Percent Squared]: 7 6 1.4
Stock Index y, z, Covariance of Return [Percent Squared]: 7 7 1.4
Stock Index y, z, Covariance of Return [Percent Squared]: 7 8 1.1
Stock Index y, z, Covariance of Return [Percent Squared]: 7 9 2.8
Stock Index y, z, Covariance of Return [Percent Squared]: 8 0 1.3
Stock Index y, z, Covariance of Return [Percent Squared]: 8 1 0.8
Stock Index y, z, Covariance of Return [Percent Squared]: 8 2 0.4
Stock Index y, z, Covariance of Return [Percent Squared]: 8 3 1.5

```
Stock Index y, z, Covariance of Return [Percent Squared]:  8 4 1.5
Stock Index y, z, Covariance of Return [Percent Squared]:  8 5 1.3
Stock Index y, z, Covariance of Return [Percent Squared]:  8 6 1.6
Stock Index y, z, Covariance of Return [Percent Squared]:  8 7 1.1
Stock Index y, z, Covariance of Return [Percent Squared]:  8 8 1.5
Stock Index y, z, Covariance of Return [Percent Squared]:  8 9 4.3
Stock Index y, z, Covariance of Return [Percent Squared]:  9 0 5.8
Stock Index y, z, Covariance of Return [Percent Squared]:  9 1 5.7
Stock Index y, z, Covariance of Return [Percent Squared]:  9 2 1.1
Stock Index y, z, Covariance of Return [Percent Squared]:  9 3 12.2
Stock Index y, z, Covariance of Return [Percent Squared]:  9 4 8.8
Stock Index y, z, Covariance of Return [Percent Squared]:  9 5 4.9
Stock Index y, z, Covariance of Return [Percent Squared]:  9 6 5.7
Stock Index y, z, Covariance of Return [Percent Squared]:  9 7 2.8
Stock Index y, z, Covariance of Return [Percent Squared]:  9 8 4.3
Stock Index y, z, Covariance of Return [Percent Squared]:  9 9 64.5

Process finished with exit code 0
```

We can also format these results in a Table 1

Table 1: Monthly returns and covariances for a portfolio of 10 investment; investments 1 through 10 correspond to tickers ZSP.TO, IVOO, VIOO, ZCN.TO, VEE.TO, VIU.TO, ZAG.TO, VBG.TO, VBU.TO, and HEP.TO

Investment	1	2	3	4	5	6	7	8	9	10
$R_i \times 100$	1.14	1.07	1.16	0.9	0.58	0.56	0.08	-0.05	0.05	0.97
$C_{ij} \times 100^2$										
1	16.2	-	-	-	-	-	-	-	-	-
2	15.6	33.2	-	-	-	-	-	-	-	-
3	16.0	35.7	41.0	-	-	-	-	-	-	-
4	14.2	19.4	20.6	18.8	-	-	-	-	-	-
5	11.5	13.1	13.2	11.9	18.6	-	-	-	-	-
6	13.5	14.6	15.0	13.6	13.6	15.8	-	-	-	-
7	2.7	2.4	1.9	2.7	2.5	2.5	2.6	-	-	-
8	1.5	1.6	1.5	1.4	1.2	1.2	1.4	2.8	-	-
9	1.3	0.8	0.4	1.5	1.5	1.3	1.6	1.1	5.8	-
10	5.8	5.7	1.1	12.2	8.8	4.9	5.7	2.8	4.3	64.5

Copy and paste the following code in the `IDE` environment. Name this code `Analysis2.py`. Note that you must have installed the `numpy` and `matplotlib` packages for this code to work.

```
#Portfolio selection to maximize return for a given level of risk
#A portfolio of 20 stocks

#Import libraries
import numpy
import matplotlib.pyplot as plt
```

```
#Stocks: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

inputFileName = "PortfolioAnalysis-10-2022-05-05.txt"

InvAmount = 10000
minRisk = 0.0010
maxRisk = 0.0030
nSimulations = 70
minRiskDesired = 0.0012
maxRiskDesired = 0.0015

dRisk = (maxRisk - minRisk) / nSimulations

nAverage = 0.0
bestRiskSum = 0.0
bestReturnSum = 0.0
X0Sum = 0.0
X1Sum = 0.0
X2Sum = 0.0
X3Sum = 0.0
X4Sum = 0.0
X5Sum = 0.0
X6Sum = 0.0
X7Sum = 0.0
X8Sum = 0.0
X9Sum = 0.0

#Load data into vectors
data = numpy.loadtxt(inputFileName)
Risk = data[:, 1]
Return = data[:, 2]
X0 = data[:, 3]
X1 = data[:, 4]
X2 = data[:, 5]
X3 = data[:, 6]
X4 = data[:, 7]
X5 = data[:, 8]
X6 = data[:, 9]
X7 = data[:, 10]
X8 = data[:, 11]
X9 = data[:, 12]
nDataset = len(data)

#plot risk versus return
#'''
```

```
plt.rc('xtick', labelsize = 16)
plt.rc('ytick', labelsize = 16)
plt.figure(figsize=(12,7))
plt.title('Scatter Plot of Risk versus Return', fontsize=20)
plt.plot(10000 * Risk, 100 * Return, 'co')
plt.xlim([0, 40])
plt.ylim([0, 2])
plt.xlabel('Risk [Percent Squared]', fontsize=20)
plt.ylabel('Monthly Return [Percent]', fontsize=20)
plt.savefig('RiskVersusReturn.png', dpi=300)
plt.show()
#'''

print('Best portfolio: Risk, Return, \n'
    'X0, X1, X2, X3, X4, X5, X6, X7, X8, X9 \n')

for j in range(0, nSimulations):
    bestReturn = -100
    for i in range(0, nDataset):
        if (Risk[i] > minRisk + j * dRisk) \
                and (Risk[i] < minRisk + (j + 1) * dRisk) \
                and (Return[i] > bestReturn):
            bestRisk = Risk[i]
            bestReturn = Return[i]
            X0Best = X0[i]
            X1Best = X1[i]
            X2Best = X2[i]
            X3Best = X3[i]
            X4Best = X4[i]
            X5Best = X5[i]
            X6Best = X6[i]
            X7Best = X7[i]
            X8Best = X8[i]
            X9Best = X9[i]

    if (bestRisk >= minRiskDesired) and (bestRisk <= maxRiskDesired):
        nAverage = nAverage + 1
        bestRiskSum = bestRiskSum + bestRisk
        bestReturnSum = bestReturnSum + bestReturn
        X0Sum = X0Sum + X0Best
        X1Sum = X1Sum + X1Best
        X2Sum = X2Sum + X2Best
        X3Sum = X3Sum + X3Best
        X4Sum = X4Sum + X4Best
        X5Sum = X5Sum + X5Best
        X6Sum = X6Sum + X6Best
```

```
        X7Sum = X7Sum + X7Best
        X8Sum = X8Sum + X8Best
        X9Sum = X9Sum + X9Best

    print('Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]: ',
        j, numpy.round(10000 * bestRisk, 1), numpy.round(100 * bestReturn, 2))
    print('X0 to X9: ', X0Best, X1Best, X2Best, X3Best, X4Best,
        X5Best, X6Best, X7Best, X8Best, X9Best)

print('AvgRisk [Percent Squared], Monthly AvgReturn [Percent]: ')
print('%0.4f %0.4f' % (10000 * bestRiskSum/nAverage, 100 * bestReturnSum/nAverage))

print('X0Avg X1Avg X2Avg X3Avg X4Avg X5Avg X6Avg X7Avg X8Avg X9Avg: ')
print('%0.3f %0.3f %0.3f %0.3f %0.3f %0.3f %0.3f %0.3f %0.3f %0.3f' %
    (X0Sum/nAverage, X1Sum/nAverage, X2Sum/nAverage, X3Sum/nAverage, X4Sum/nAverage,
    X5Sum/nAverage, X6Sum/nAverage, X7Sum/nAverage, X8Sum/nAverage, X9Sum/nAverage))

print('Inv0 Inv1 Inv2 Inv3 Inv4 Inv5 Inv6 Inv7 Inv8 Inv9: ')
print('%5.0f %5.0f %5.0f %5.0f %5.0f %5.0f %5.0f %5.0f %5.0f %5.0f'
    % (X0Sum/nAverage*InvAmount, X1Sum/nAverage*InvAmount,
    X2Sum/nAverage*InvAmount, X3Sum/nAverage*InvAmount, X4Sum/nAverage*InvAmount,
    X5Sum/nAverage*InvAmount, X6Sum/nAverage*InvAmount, X7Sum/nAverage*InvAmount,
    X8Sum/nAverage*InvAmount, X9Sum/nAverage*InvAmount))
```

This code divides the risk domain into `nSimulations` from `minRisk` to `maxRisk`. It also takes a `minRiskDesired` and `maxRiskDesired`. It subsequently searches for the maximum return in each risk interval for the desired range of risk. In the end it averages the risk, return, and the weightings of investments for the desired range of risk. The following console output can be generated upon successful execution of this code.

```
Best portfolio: Risk, Return,
X0, X1, X2, X3, X4, X5, X6, X7, X8, X9

Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  0 10.1 0.89
X0 to X9:  0.6 0.1 0.0 0.0 0.0 0.0 0.0 0.1 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  1 10.4 0.89
X0 to X9:  0.5 0.0 0.1 0.1 0.0 0.0 0.1 0.0 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  2 10.8 0.9
X0 to X9:  0.5 0.1 0.1 0.0 0.0 0.0 0.1 0.0 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  3 11.1 0.91
X0 to X9:  0.5 0.1 0.1 0.0 0.0 0.0 0.2 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  4 11.3 0.9
X0 to X9:  0.6 0.1 0.0 0.1 0.0 0.0 0.2 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  5 11.6 0.91
X0 to X9:  0.6 0.0 0.0 0.0 0.2 0.0 0.1 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  6 11.9 0.94
```

```
X0 to X9:  0.5 0.0 0.1 0.1 0.1 0.0 0.0 0.0 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  7 12.3 0.94
X0 to X9:  0.5 0.2 0.0 0.0 0.1 0.0 0.0 0.0 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  8 12.4 0.99
X0 to X9:  0.6 0.0 0.1 0.1 0.0 0.0 0.0 0.0 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  9 12.8 1.01
X0 to X9:  0.6 0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  10 13.0 1.02
X0 to X9:  0.6 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  11 13.4 1.01
X0 to X9:  0.7 0.0 0.1 0.1 0.0 0.0 0.0 0.0 0.1 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  12 13.7 1.0
X0 to X9:  0.5 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.1 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  13 14.0 1.01
X0 to X9:  0.6 0.0 0.1 0.0 0.1 0.1 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  14 14.1 1.0
X0 to X9:  0.4 0.0 0.3 0.0 0.0 0.0 0.0 0.0 0.1 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  15 14.5 1.02
X0 to X9:  0.5 0.0 0.1 0.2 0.1 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  16 14.8 1.06
X0 to X9:  0.6 0.1 0.1 0.0 0.1 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  17 14.9 1.05
X0 to X9:  0.5 0.0 0.2 0.0 0.1 0.0 0.0 0.0 0.0 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  18 15.4 1.11
X0 to X9:  0.6 0.0 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  19 15.5 1.09
X0 to X9:  0.6 0.1 0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  20 15.9 1.06
X0 to X9:  0.5 0.1 0.2 0.0 0.1 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  21 16.3 1.07
X0 to X9:  0.7 0.2 0.0 0.0 0.0 0.1 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  22 16.3 1.07
X0 to X9:  0.5 0.0 0.3 0.0 0.0 0.1 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  23 16.6 1.08
X0 to X9:  0.4 0.1 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  24 17.0 1.07
X0 to X9:  0.6 0.2 0.1 0.0 0.1 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  25 17.3 1.1
X0 to X9:  0.5 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  26 17.7 1.11
X0 to X9:  0.5 0.2 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  27 18.0 1.11
X0 to X9:  0.4 0.0 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  28 18.0 1.09
X0 to X9:  0.4 0.2 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  29 18.3 1.1
```

```
X0 to X9:  0.4 0.1 0.3 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  30 18.7 1.11
X0 to X9:  0.4 0.0 0.4 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  31 19.1 1.09
X0 to X9:  0.3 0.3 0.2 0.0 0.0 0.0 0.0 0.0 0.0 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  32 19.4 1.11
X0 to X9:  0.5 0.2 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  33 19.6 1.11
X0 to X9:  0.4 0.2 0.3 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  34 19.7 1.08
X0 to X9:  0.3 0.3 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  35 20.1 1.09
X0 to X9:  0.3 0.2 0.3 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  36 20.5 1.1
X0 to X9:  0.3 0.1 0.4 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  37 20.7 1.09
X0 to X9:  0.4 0.4 0.1 0.1 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  38 21.0 1.09
X0 to X9:  0.3 0.5 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  39 21.4 1.09
X0 to X9:  0.2 0.3 0.3 0.0 0.0 0.0 0.0 0.0 0.0 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  40 21.6 1.11
X0 to X9:  0.3 0.3 0.3 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  41 21.7 1.1
X0 to X9:  0.2 0.2 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  42 22.1 1.11
X0 to X9:  0.2 0.1 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.2
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  43 22.6 1.12
X0 to X9:  0.3 0.1 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  44 22.7 1.09
X0 to X9:  0.2 0.2 0.4 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  45 23.0 1.07
X0 to X9:  0.3 0.3 0.3 0.0 0.0 0.1 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  46 23.3 1.08
X0 to X9:  0.2 0.6 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  47 23.7 1.1
X0 to X9:  0.3 0.3 0.3 0.1 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  48 23.8 1.13
X0 to X9:  0.4 0.2 0.4 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  49 24.0 1.1
X0 to X9:  0.2 0.4 0.3 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  50 24.5 1.09
X0 to X9:  0.1 0.0 0.6 0.2 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  51 24.7 1.1
X0 to X9:  0.3 0.6 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  52 24.9 1.05
```

```
X0 to X9:  0.1 0.3 0.3 0.3 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  53 25.2 1.08
X0 to X9:  0.1 0.3 0.4 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  54 25.6 1.09
X0 to X9:  0.2 0.5 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  55 25.7 1.09
X0 to X9:  0.1 0.2 0.5 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  56 26.2 1.09
X0 to X9:  0.2 0.4 0.3 0.1 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  57 26.3 1.1
X0 to X9:  0.1 0.1 0.6 0.1 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  58 26.8 1.1
X0 to X9:  0.2 0.3 0.4 0.1 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  59 26.9 1.06
X0 to X9:  0.1 0.2 0.5 0.1 0.1 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  60 27.4 1.11
X0 to X9:  0.2 0.2 0.5 0.1 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  61 27.5 0.99
X0 to X9:  0.0 0.3 0.5 0.1 0.0 0.0 0.0 0.0 0.1 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  62 27.8 1.11
X0 to X9:  0.1 0.3 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  63 28.0 0.98
X0 to X9:  0.2 0.0 0.1 0.0 0.0 0.1 0.0 0.0 0.0 0.6
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  64 28.3 1.06
X0 to X9:  0.0 0.2 0.6 0.0 0.0 0.1 0.0 0.0 0.0 0.1
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  65 28.7 0.92
X0 to X9:  0.0 0.2 0.1 0.0 0.0 0.0 0.1 0.0 0.0 0.6
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  66 28.9 1.06
X0 to X9:  0.0 0.2 0.5 0.3 0.0 0.0 0.0 0.0 0.0 0.0
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  67 29.4 1.02
X0 to X9:  0.1 0.0 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.6
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  68 29.7 1.02
X0 to X9:  0.1 0.2 0.1 0.0 0.0 0.0 0.0 0.0 0.0 0.6
Simulation, Best Risk [Percent Squared], Best Monthly Return [Percent]:  69 30.0 0.94
X0 to X9:  0.0 0.2 0.0 0.1 0.0 0.1 0.0 0.0 0.0 0.6
AvgRisk [Percent Squared], Monthly AvgReturn [Percent]:
13.6245 1.0117
X0Avg X1Avg X2Avg X3Avg X4Avg X5Avg X6Avg X7Avg X8Avg X9Avg:
0.555 0.055 0.127 0.036 0.045 0.009 0.000 0.000 0.064 0.109
Inv0 Inv1 Inv2 Inv3 Inv4 Inv5 Inv6 Inv7 Inv8 Inv9:
5545   545  1273   364   455    91     0     0   636  1091

Process finished with exit code 0
```

The code also draws the `RiskVersusReturn.png` figure and saves it in the same directory. Figure 2 shows the risk versus return for the 500,000 portfolios of investments.

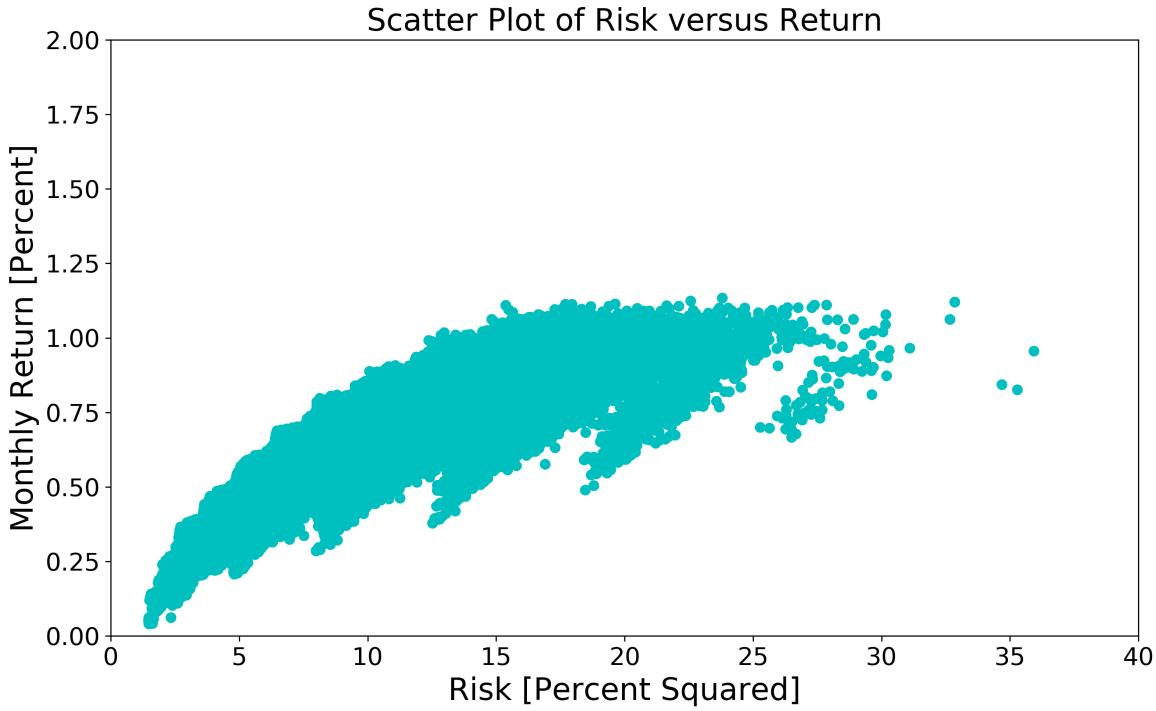

Figure 2: Risk versus return of 500,000 portfolios generated randomly based on 10 investments

For this specific example, this portfolio is shown in Table 2. The table shows the weights X_i and the investment allocation I_i associated with each weight if the investor invests I =\$10,000 in total.

Table 2: Portfolio weights X_i and investment amounts I_i assuming a total investment of I =\$10,000 in the portfolio; investments 1 through 10 correspond to tickers ZSP.TO, IVOO, VIOO, ZCN.TO, VEE.TO, VIU.TO, ZAG.TO, VBG.TO, VBU.TO, and HEP.TO

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
Weights	0.555	0.055	0.127	0.036	0.045	0.009	0.000	0.000	0.064	0.109
	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
[\$]	5545	545	1273	364	455	91	0	0	636	1091

3 Task Activity

Now that you can successfully run the python codes and generate the results. Perform the following analyses by manipulating the python codes `Analysis1.py`, `Analysis2.py` and generating a new text file for historical sample returns of 10 other investments.

Pick a 6 year period that ends at the present date. Pick 10 investment tickers and obtain their historical share prices and annual dividends. Construct an investment portfolio. Generate the correlation matrix and the risk versus return map. Show the correlation matrix and the map. Identify minimum and maximum desired risks and arrive at a suitable portfolio of investments. Report the average risk, return, and investment weights.

Try to answer the following questions:

1. How did you decide which investment tickers to include?
2. Which sources did you use to find share prices and dividends?
3. What is a realistic range for minimum and maximum desired risk, i.e. `minRiskDesired` and `maxRiskDesired`?
4. What are the suitable values for variables `nSimulations`, `minRisk`, and `maxRisk`? Does your simulation crash if these values are not appropriate?
5. What are the average risk, return, and investment weights for your portfolio?

Hint: you may have to find the suitable values for variables `minRiskDesired`, `maxRiskDesired`, `nSimulations`, `minRisk`, and `maxRisk` by trial and error.

4 Reporting Guidelines

Please provide your report as a single PDF file. Do not submit your python code or data set as separate attachments. Instead, list your python code in the body of your report in the Appendix. The Microsoft Word format will not be accepted. Your report may include the following sections: Introduction, Methodology, Results and Discussions, Conclusion, and Appendix.

I *strongly discourages* use of Microsoft Excel for technical communications. You should perform any data analysis or plotting using Python. If you use Microsoft Excel in any part of your analysis or reporting, your report will not be graded.

Students are *strongly encouraged* if they use LaTeX(instead of Microsoft Word) to generate the report. I will give two bonus marks for using LaTeX. I generally suggest the open source Text Studio for production of reports, manuscripts, and theses in LaTeX. The program can be downloaded via `https://www.texstudio.org/`.

References

[Fabozzi and Jones, 2019] Fabozzi, F. J. and Jones, F. J. (2019). *Foundations of Global Financial Markets in Institutions*. MIT Press, Cambridge, 5th edition.

[Markowitz, 1952] Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1):77–91.

[Markowitz, 1959] Markowitz, H. (1959). *Portfolio Selection: Efficient Diversification of Investments*. John Wiley & Sons Inc., New York.