A Project Report

On

Optimization Methods in Finance

Submitted in partial fulfilment of the requirement for the degree of

Bachelor of Technology

In

Electronics and Communication Engineering

Submitted By: **Under the Supervision of:**

Apoorva Anand (17102089) Dr. Ankit Garg

Department of Electronics and Communication Engineering

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY,

NOIDA

May, 2020

TABLE OF CONTENTS

1. Certificate -- i
2. Acknowledgement -- ii
3. Chapter 1 – INTRODCUTION -- 1
4. Chapter 2 – TECHNICAL DESCRIPTION --- 6
5. Chapter 3 – CODE -- 15
6. Chapter 4 – FUTURE SCOPE -- 27
7. Chapter 5 – CONCLUSION ---28
8. References --29

CERTIFICATE

It is certified that the work contained in the project report titled " **Optimization Method in Finance**" by "**Apoorva Anand**" has been carried out under my supervision and this work has not been submitted elsewhere for a degree.

Signature of Supervisor

Dr. Ankit Garg

ECE

JIIT NOIDA

May,2020

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude and deep regards to my mentor Dr. Ankit Garg for the continuous support in the B.Tech Minor Project, for his patience, motivation, enthusiasm and immense knowledge. I owe the success of this project to him. His guidance helped me in all time of my project and writing of report. I have to thank my external Mr. Shivaji Tyagi in my intermediate project evaluation and presentation for his encouragement, insightful comments and hard questions.

Also would like to extend the gratefulness to my parents and family for the constant love and support.

Signature of Students

Apoorva Anand

Chapter 1 – INTRODUCTION

In the face of risky investment opportunities, investment options cannot rely solely on the expected profit rate. Investors should be willing to bear the high risk if it expects to earn high rates of return . Therefore, the stock price valuation analysis, the selection of a combination of stocks in a portfolio optimum, and measure the risk of investing in the capital market is very necessary. This is where the mathematical model has a very important role to analyze the formation of the portfolio.

The mathematical model seeks to facilitate the analysis of investment. This is because in a single index model assumed that the correlations of returns between stocks occur due to the stock response to changes in the Composite Stock Price Index (general market index).When the market improves, the price of individual stocks also increased, and vice versa . Single index model is used for valuation of stocks in an investment. This research examines the optimal strategy for investing in the stocks indices of emerging markets. Trading strategies are set regarding different technical indicators based on moving averages and volatility of the value and returns on stock indices.

1.1 Financial Engineering

Financial engineering is the use of mathematical techniques to solve financial problems. Financial engineering uses tools and knowledge from the fields of computer science, electrical engineering, statistics, economics, and applied mathematics to address current financial issues as well as to devise new and innovative financial products.

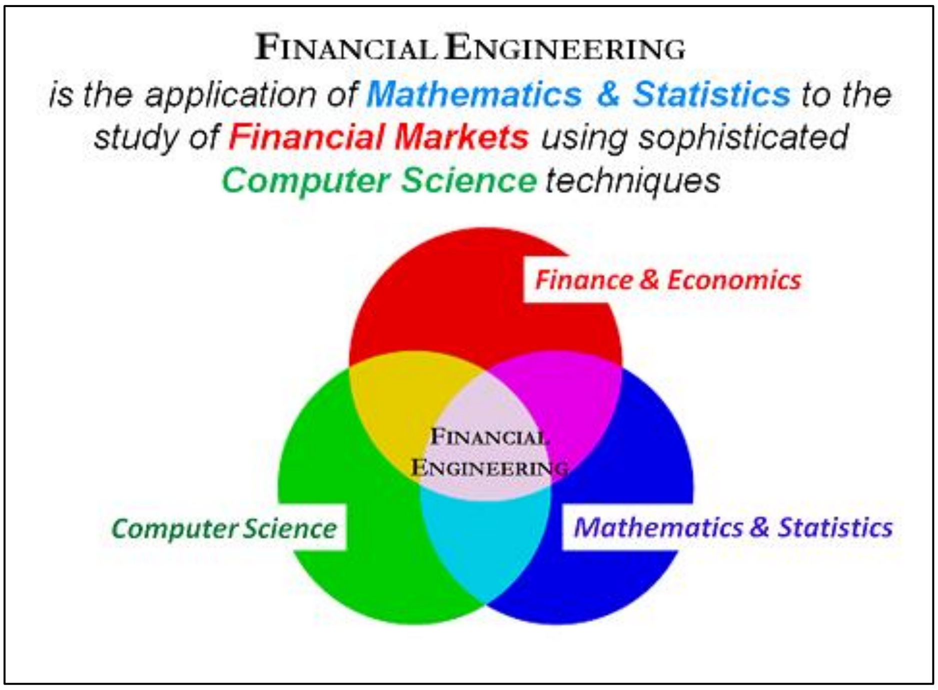
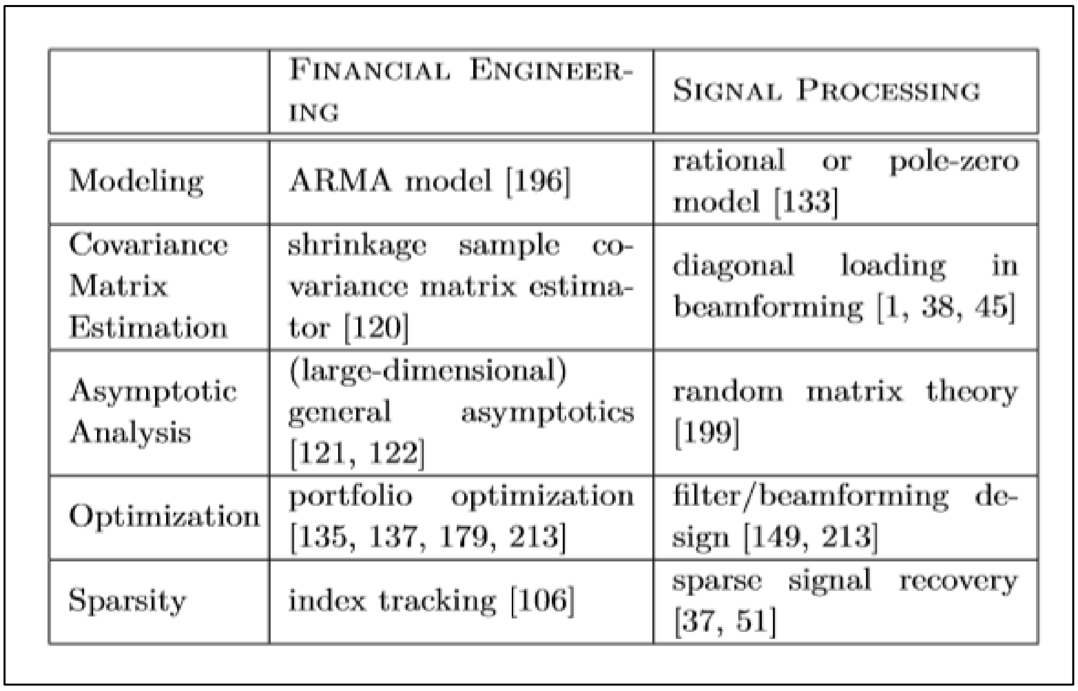

FINANCIAL ENGINEERING
is the application of Mathematics & Statistics to the study of Financial Markets using sophisticated Computer Science techniques
Finance & Economics
FINANCIAL ENGINEERING
Computer Science
Mathematics & Statistics

Fig 1. Financial Engineering

1.2 A Signal Processing Perspective on Financial Engineering

Despite the different natures of financial engineering and electrical engineering, both areas are intimately connected on a mathematical level. The foundations of financial engineering lie on the statistical analysis of numerical time series and the modeling of the behavior of the financial markets in order to perform predictions and systematically optimize investment strategies. Similarly, the foundations of electrical engineering, for instance, wireless communication systems, lie on statistical signal processing and the modeling of communication channels in order to perform predictions and systematically optimize transmission strategies.

Thus, this monograph is about investment in financial assets treated as a signal processing and optimization problem



	Financial Engineering	Signal Processing
Modeling	ARMA model [196]	rational or pole-zero model [133]
Covariance Matrix Estimation	shrinkage sample covariance matrix estimator [120]	diagonal loading in beamforming [1, 38, 45]
Asymptotic Analysis	(large-dimensional) general asymptotics [121, 122]	random matrix theory [199]
Optimization	portfolio optimization [135, 137, 179, 213]	filter/beamforming design [149, 213]
Sparsity	index tracking [106]	sparse signal recovery [37, 51]

Fig 2. Connections between Financial Engineering and Signal Processing

1.3 Portfolio Optimization

Portfolio:-A portfolio is a grouping of financial assets such as stocks, bonds, commodities, currencies and cash equivalents, as well as their fund counterparts. Investors should construct an investment portfolio in accordance with their risk tolerance and investing objectives

Optimization:- An optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function. Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics, and the development of solution methods has been of interest in mathematics for centuries

The portfolio optimization plays an important role in determining the portfolio strategies for investors. What investors hope to achieve through a portfolio optimization is to maximize portfolio return and minimize portfolio risk. Since the return changes based on risk investors have to balance the contradiction between risk and return for their investment. There is therefore no single optimized portfolio to satisfy all investors. The optimal portfolio is determined by the preferences of the investor’s risk and return .

1.4 Robust Portfolio Optimization Model

Markowitz’s mean-variance analysis sets the basis for modern portfolio optimization theory . However, the mean-variance analysis has been criticized for being sensitive to estimation errors in the mean and covariance matrix of the asset returns. Compared to the covariance matrix, the mean of the asset returns is more influential and harder to estimate. Therefore, many studies focus on the global minimum variance (GMV) formulation, which only involves estimating the covariance matrix of the asset returns. Estimating the covariance matrix of asset returns is challenging due to the high dimensionality and heavy-tailedness of asset return data. Specifically, the number of assets under management is usually much larger than the sample size of exploitable historical data. On the other hand, extreme events are typical in financial asset prices, leading to heavy-tailed asset returns. To overcome the curse of dimensionality, structured covariance matrix estimators are proposed for asset return data considered estimators based on factor models with observable factors. studied covariance matrix estimators based on latent factor models proposed to shrink the sample covariance matrix towards highly structured covariance matrices, including the identity matrix, order 1 autoregressive covariance matrices, and one-factor-based covariance matrix estimators. These estimators are commonly based on the sample covariance matrix. Gaussian tail assumptions are required to guarantee consistency. For heavy-tailed data, robust estimators of covariance matrices are desired. Classic robust covariance matrix estimators include M-estimators, minimum volume ellipsoid (MVE) and minimum covari1 ance determinant (MCD) estimators, S-estimators, and estimators based on data outlyingness and depth . These estimators are specifically designed for data with very low dimensions and large sample sizes

Many investors use optimization to determine their optimal investment portfolio. Unfortunately, optimal portfolios are sensitive to changing input parameters, i.e., they are not robust. Traditional robust optimization approaches aim for an optimal and robust portfolio which, ideally, is the final investment decision. In practice, however, portfolio optimization supports but seldomly replaces the investment decision process.

Robust optimization approaches work well when all market information is quantified and incorporated in the optimization problem. But, despite eorts to incorporate information such as transaction costs, expert opinion and liquidity, an optimization problem remains a simplification of reality. Therefore, in practice, investors combine the optimal portfolio with additional information that was not or could not be incorporated. So, for investors, portfolio optimization is a tool that supports but does not replace their decision process.

1.5 Constraint Equation for Optimizing Weights

Stock price assessment, selection of optimum combination, and measure the risk of a portfolio investment is one important issue for investors. In this research different constraint and equations are generated to get the optimized weights to invest for in a specific user defined ways.

With the advent of digital technology and the accompanying gains in processing speed and data storage, techniques in signal processing have become increasingly sought after in the finance industry. Finance is one such field since financial data is very often compiled with reference to time as the independent variable. More specifically, most forms of time varying financial data may be interpreted as discrete time signals. Not surprisingly, mathematical techniques used in digital signal processing are at present making forays into the finance industry precisely because the problems relating to signals faced by professionals in either field are identical for signals, whether electrical or financial, receive the same treatment in the abstract realm of signal processing.

So, using the same techniques of signal processing, similar mathematics is used for the optimization of stock values. Some of the constraints from this research are:-

i. $ax+by=c$ given $a+b=1$ based on maximum return as well as minimum variance
ii. $ax+by+cz+dw=const$ given $a+b+c+d=1$ based on maximum return as well as minimum variance.
iii. $ax+by+bz+bw=const$ given $a+3b=1$ based on maximum return as well as minimum variance
iv. $ax+by+cz+dw=const$ given $a+b+c+d=1$ and $b:c:d=0.5:0.3:0.2(1-a)$ based on maximum return as well as minimum variance

Chapter 2 – TECHNICAL DESCRIPTION

2.1 Software Used

I. JUPYTER NOTEBOOK

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook interface

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

IPython

ØMQ

Tornado (web server)

jQuery

Bootstrap (front-end framework)

MathJax

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for many programming languages, including Python, R, Julia and Haskell.

Fig 3. Jupyter Notebook

II . NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Fig 4.Numpy Module

III. Pandas

“pandas” is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy andintuitive.It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet
- Ordered and unordered (not necessarily fixed-frequency) time series data.
- Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels
- Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's data frame provides and much more. "pandas" is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Here are just a few of the things that "pandas" does well:

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data
- Size mutability: columns can be inserted and deleted from DataFrame and higher dimensional objects
- Automatic and explicit data alignment: objects can be explicitly aligned to a set of labels, or the user can simply ignore the labels and let Series, DataFrame, etc. automatically align the data for you in computations
- Powerful, flexible group by functionality to perform split-apply-combine operations on data sets, for both aggregating and transforming data
- Make it easy to convert ragged, differently-indexed data in other Python and NumPy data structures into DataFrame objects
- Intelligent label-based slicing, fancy indexing, and subsetting of large data sets
- Intuitive merging and joining data sets

- Flexible reshaping and pivoting of data sets
- Hierarchical labeling of axes (possible to have multiple labels per tick)
- Robust IO tools for loading data from flat files (CSV and delimited), Excel files, databases, and saving / loading data from the ultrafast HDF5 format
- Time series-specific functionality: date range generation and frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging, etc.

Many of these principles are here to address the shortcomings frequently experienced using other languages / scientific research environments. For data scientists, working with data is typically divided into multiple stages: munging and cleaning data, analyzing / modeling it, then organizing the results of the analysis into a form suitable for plotting or tabular display. “pandas” is the ideal tool for all of these tasks.

Fig 5. Pandas Module

IV. Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community,and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.

Fig 6. Matplotlib logo

V. Seaborn

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Here is some of the functionality that seaborn offers:

A dataset-oriented API for examining relationships between multiple variables

- Specialized support for using categorical variables to show observations or aggregate statistics
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- Automatic estimation and plotting of linear regression models for different kinds dependent variables
- Convenient views onto the overall structure of complex datasets
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations
- Concise control over matplotlib figure styling with several built-in themes
- Tools for choosing color palettes that faithfully reveal patterns in your data

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on dataframes and array containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

2.2. Parameters Used

I. ROBUST MEAN-VARIANCE PORTFOLIO MODEL

The mean-variance portfolio choice problem chooses w to maximize

$E[w(\vec{r} - r_f\mathbf{1})]] - var[w(\vec{r} - r_f\mathbf{1})]$

which equals, $w'\mu - (\delta/2)w'\Sigma w$

A robust decision maker can be modelled as replacing the mean return $E[w(\vec{r} - rf1)]E[w(r \rightarrow -rf1)]$ with the risk-sensitive

$T[w(\vec{r} - r_f\mathbf{1})] = w'\mu - (1/2\theta)w'\Sigma w$

that comes from replacing the mean μ of $\vec{r}$ − r_f1 with the worst-case mean

$\mu - \theta^{-1}\Sigma w$

Notice how the worst-case mean vector depends on the portfolio w.

The operator T is the indirect utility function that emerges from solving a problem in which an agent who chooses probabilities does so in order to minimize the expected utility of a maximizing .

The robust version of the mean-variance portfolio choice problem is then to choose a portfolio w that maximizes

$T[w(\vec{r} - r_f\mathbf{1})] - (\delta/2)w'\Sigma w$

or

$w'(\mu - \theta^{-1}\Sigma w) - (\delta/2)w'\Sigma w$

The minimiser is

$w_{rob} = (1/\delta+\gamma)\Sigma^{-1}\mu$

where $\gamma \equiv \theta^{-1}$ is sometimes called the risk-sensitivity parameter.

An increase in the risk-sensitivity parameter γ shrinks the portfolio weights toward zero in the same way that an increase in risk aversion does.

We want to illustrate the "folk theorem" that with high or moderate frequency data, it is more difficult to estimate means than variances.

In order to operationalize this statement, we take two analog estimators:

- sample average: X^{-}_{N}=(1/N)$\sum_{i=1}^{N} Xi$
- sample variance: S_N=(1/N−1)$\sum_{t=1}^{N}$ $(X_i-X^{-}_{N})^2$

to estimate the unconditional mean and unconditional variance of the random variable X, respectively.

To measure the "difficulty of estimation", we use *mean squared error* (MSE), that is the average squared difference between the estimator and the true value.

Assuming that the process{Xi}is ergodic, both analog estimators are known to converge to their true values as the sample size N goes to infinity.

More precisely for all ε>0

$$\lim_{N\to\infty} P\left\{|\bar{X}_N - \mathbb{E}X| > \varepsilon\right\} = 0$$

$$\lim_{N\to\infty} P\{|S_N - \mathbb{V}X| > \varepsilon\} = 0$$

A necessary condition for these convergence results is that the associated MSEs vanish as NN goes to infinity, or in other words,

$$\mathbf{MSE}(\bar{X}_N, \mathbb{E}X) = o(1) \qquad \mathbf{and} \qquad \mathbf{MSE}(S_N, \mathbb{V}X) = o(1)$$

Even if the MSEs converge to zero, the associated rates might be different. Looking at the limit of the *relative MSE* (as the sample size grows to infinity)

$$\frac{\mathrm{MSE}(S_N, \mathbb{V}X)}{\mathrm{MSE}(\bar{X}_N, \mathbb{E}X)} = \frac{o(1)}{o(1)} \underset{N\to\infty}{\longrightarrow} B$$

can inform us about the relative (asymptotic) rates.

We will show that in general, with dependent data, the limit BB depends on the sampling frequency.

In particular, we find that the rate of convergence of the variance estimator is less sensitive to increased sampling frequency than the rate of convergence of the mean estimator.

Hence, we can expect the relative asymptotic rate, BB, to get smaller with higher frequency data, illustrating that "it is more difficult to estimate means than variances".

That is, we need significantly more data to obtain a given precision of the mean estimate than for our variance estimate.

II. Constraint Equation for Optimizing Weights

1. Rate of Return

$$r_t = \frac{Pt - Pt{-}1}{Pt{-}1}$$

2. Expected Return

$$\mu_i = E(r^i) = \frac{\sum_{t=1}^{m} r_t^i}{m}$$

3. Variance and standard deviation

$$\sigma_i^{\,2} = \mathrm{Var}(r^i) = \frac{\sum_{t=1}^{m}(r_t^i - \mu_i)^2}{m-1}$$

$$\sigma_i = \sqrt{\sigma_i^2} = \sqrt{\frac{\sum_{t=1}^{m}(r_t^i - \mu_i)^2}{m-1}}$$

4. Covariance

$$\sigma_{ij} = Cov(r^i, r^j) = \frac{\sum_{t=1}^{m}(r_i^t - \mu_i)(r_j^t - \mu_j)}{m}$$

5. Minimizing variance according to different constraints

Every possible asset combination can be plotted in variance space, and the collection of all such possible portfolios defines a region in this space. The portfolio at which the variance is minimum is the optimized portfolio.

$$\begin{aligned}&\min \sigma_P = \min \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} \omega_i \omega_j \sigma_{ij}}\ , \\ &0 \leq \omega_i \leq 1, \quad i = 1, \ldots, n \\ &\sum_{i=1}^{n} \omega_i = 1\end{aligned}$$

Where:

ω_i is a percentage of capital that will be invested in asset i;

r_i is the return on asset i;

μ_i the expected return on asset i;

μ_{ij} is the covariance between the return on assets i and j;

$E(r_p)$ is the expected return of the portfolio;

σ_p is the risk of the portfolio

Chapter 3 – CODE

1. <u>ROBUST MEAN VARIANCE OPTIMIZATION MODEL</u>

A Special Case – IID Sample

We start our analysis with the benchmark case of IID data. Consider a sample of size N generated by the following IID process,

$$X_i \sim N(\mu,\sigma 2)$$

the MSE is

$$MSE(\overline{X}_N,\mu)=\sigma^2/N$$

$$MSE(S_N,\sigma^2)=2\sigma^4/N-1$$

Both estimators are unbiased and hence the MSEs reflect the corresponding variances of the estimators.

Furthermore, both MSEs are o(1) with a (multiplicative) factor of difference in their rates of convergence:

$$\frac{\mathrm{MSE}(S_N,\sigma^2)}{\mathrm{MSE}(\bar{X}_N,\mu)} = \frac{N2\sigma^2}{N-1} \quad \underset{N\to\infty}{\to} \quad 2\sigma^2$$

To investigate how sampling frequency affects relative rates of convergence, we assume that the data are generated by a mean-reverting continuous time process of the form

$$dX_t = -\kappa(X_t-\mu)\,dt + \sigma\,dW_t$$

where μ is the unconditional mean, $\kappa>0$ is a persistence parameter, and $\{Wt\}$ is a standardized Brownian motion.

Observations arising from this system in particular discrete periods $T(h)\equiv\{nh:n\in Z\}$ with $h>0$ can be described by the following process

$$X_{t+1}=(1-\exp(-\kappa h))\mu+\exp(-\kappa h)X_t+\varepsilon_{t,h}$$

$\varepsilon_{t,h} \sim N(0,\Sigma_h)$ with $\Sigma_h=\sigma^2(1-\exp(-2\kappa h)) / 2\kappa$

Straightforward calculations show that the autocorrelation function for the stochastic process $\{X_t\}_{t\in T(h)}$ is

$\Gamma_h(n)\equiv \mathrm{corr}(X_{t+hn},X_t)=\exp(-\kappa hn)$

auto-covariance function is

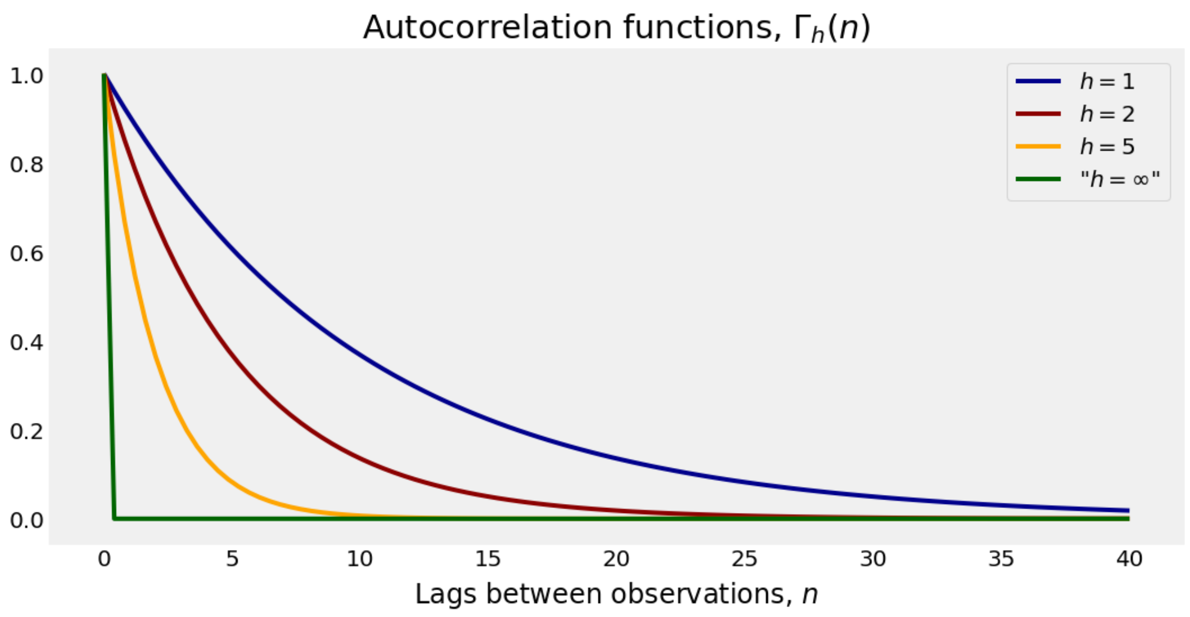
$\gamma_h(n)\equiv \mathrm{cov}(X_{t+hn},X_t)=\exp(-\kappa hn)\sigma^2/2\kappa.$

Now, importing the libraries and moving further,

```
import numpy as np
import scipy as sp
import scipy.stats as stat
import matplotlib.pyplot as plt
%matplotlib inline
from ipywidgets import interact, FloatSlider
```

```
μ = .0
κ = .1
σ = .5
var_uncond = σ**2 / (2 * κ)

n_grid = np.linspace(0, 40, 100)
autocorr_h1 = np.exp(-κ * n_grid * 1)
autocorr_h2 = np.exp(-κ * n_grid * 2)
autocorr_h5 = np.exp(-κ * n_grid * 5)
autocorr_h1000 = np.exp(-κ * n_grid * 1e8)

fig, ax = plt.subplots(figsize=(8, 4))
ax.plot(n_grid, autocorr_h1, label=r'$h=1$', c='darkblue', lw=2)
ax.plot(n_grid, autocorr_h2, label=r'$h=2$', c='darkred', lw=2)
ax.plot(n_grid, autocorr_h5, label=r'$h=5$', c='orange', lw=2)
ax.plot(n_grid, autocorr_h1000, label=r'"$h=\infty$"', c='darkgreen', lw=2)
ax.legend()
ax.grid()
ax.set(title=r'Autocorrelation functions, $\Gamma_h(n)$',
       xlabel=r'Lags between observations, $n$')
plt.show()
```

We get this graph,



The following figure illustrates how the dependence between the observations is related to the sampling frequency

- For any given h, the autocorrelation converges to zero as we increase the distance – n– between the observations. This represents the "weak dependence" of the $\overline{X}$ process.
- Moreover, for a fixed lag length, n, the dependence vanishes as the sampling frequency goes to infinity. In fact, letting h go to ∞ gives back the case of IID data.

Frequency and the Mean Estimator

Consider again the AR(1) process generated by discrete sampling with frequency h. Assume that we have a sample of size N and we would like to estimate the unconditional mean – in our case the true mean is μ.

Again, the sample average is an unbiased estimator of the unconditional mean

$$\mathbb{E}[\bar{X}_N] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}[X_i] = \mathbb{E}[X_0] = \mu$$

The variance of the sample mean is given by

$$
\begin{aligned}
\mathbb{V}\left(\bar{X}_N\right) &= \mathbb{V}\left(\frac{1}{N}\sum_{i=1}^{N} X_i\right) \\
&= \frac{1}{N^2}\left(\sum_{i=1}^{N}\mathbb{V}(X_i) + 2\sum_{i=1}^{N-1}\sum_{s=i+1}^{N}\operatorname{cov}(X_i, X_s)\right) \\
&= \frac{1}{N^2}\left(N\gamma(0) + 2\sum_{i=1}^{N-1} i\cdot\gamma(h\cdot(N-i))\right) \\
&= \frac{1}{N^2}\left(N\frac{\sigma^2}{2\kappa} + 2\sum_{i=1}^{N-1} i\cdot\exp(-\kappa h(N-i))\frac{\sigma^2}{2\kappa}\right)
\end{aligned}
$$

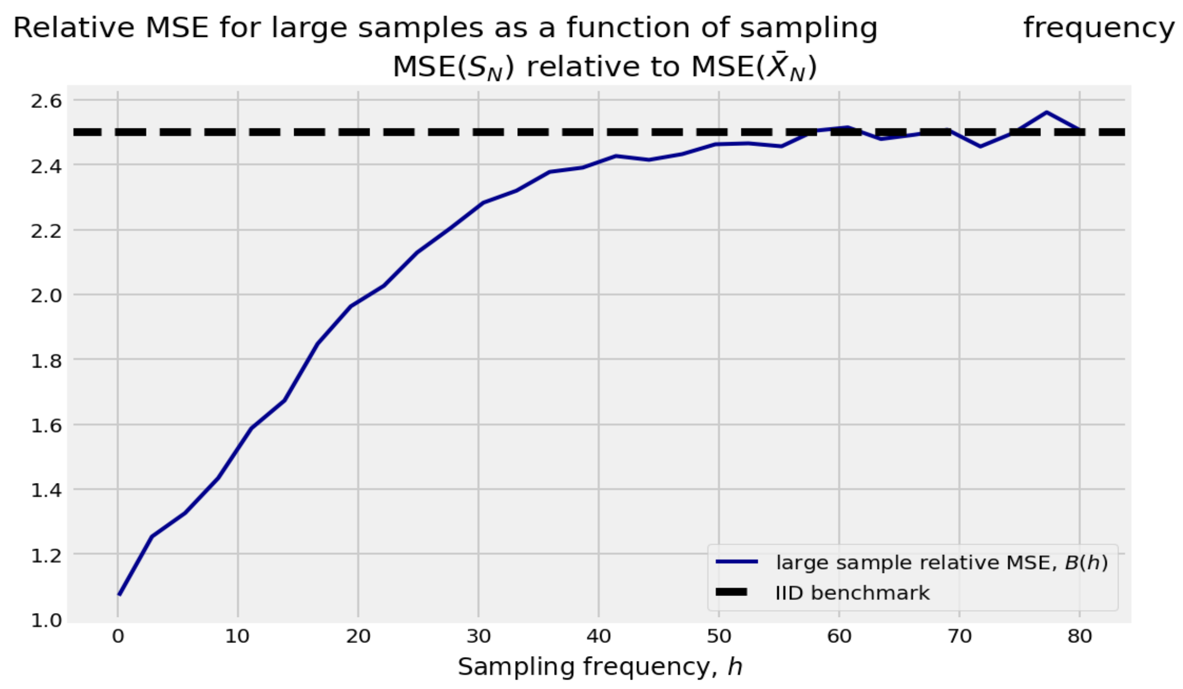
Moving on,

```
def sample_generator(h, N, M):
    φ = (1 - np.exp(-κ * h)) * μ
    ρ = np.exp(-κ * h)
    s = σ**2 * (1 - np.exp(-2 * κ * h)) / (2 * κ)

    mean_uncond = μ
    std_uncond = np.sqrt(σ**2 / (2 * κ))

    ε_path = stat.norm(0, np.sqrt(s)).rvs((M, N))

    y_path = np.zeros((M, N + 1))
    y_path[:, 0] = stat.norm(mean_uncond, std_uncond).rvs(M)

    for i in range(N):
        y_path[:, i + 1] = φ + ρ * y_path[:, i] + ε_path[:, i]

    return y_path
```

```
N_app, M_app = 1000, 30000          # Sample size, number of simulations
h_grid = np.linspace(.1, 80, 30)

var_est_store = []
mean_est_store = []
labels = []

for h in h_grid:
    labels.append(h)
    sample = sample_generator(h, N_app, M_app)
    mean_est_store.append(np.mean(sample, 1))
    var_est_store.append(np.var(sample, 1))
```

```python
for h in h_grid:
    labels.append(h)
    sample = sample_generator(h, N_app, M_app)
    mean_est_store.append(np.mean(sample, 1))
    var_est_store.append(np.var(sample, 1))

var_est_store = np.array(var_est_store)
mean_est_store = np.array(mean_est_store)

# Save mse of estimators
mse_mean = np.var(mean_est_store, 1) + (np.mean(mean_est_store, 1) - μ)**2
mse_var = np.var(var_est_store, 1) \
          + (np.mean(var_est_store, 1) - var_uncond)**2

benchmark_rate = 2 * var_uncond       # IID case

# Relative MSE for large samples
rate_h = mse_var / mse_mean

fig, ax = plt.subplots(figsize=(8, 5))
ax.plot(h_grid, rate_h, c='darkblue', lw=2,
        label=r'large sample relative MSE, $B(h)$')
ax.axhline(benchmark_rate, c='k', ls='--', label=r'IID benchmark')
ax.set_title('Relative MSE for large samples as a function of sampling \
              frequency \n MSE($S_N$) relative to MSE($\\bar X_N$)')
ax.set_xlabel('Sampling frequency, $h$')
ax.legend()
plt.show()
```

And, we get this graph



The above figure illustrates the relationship between the asymptotic relative MSEs and the sampling frequency

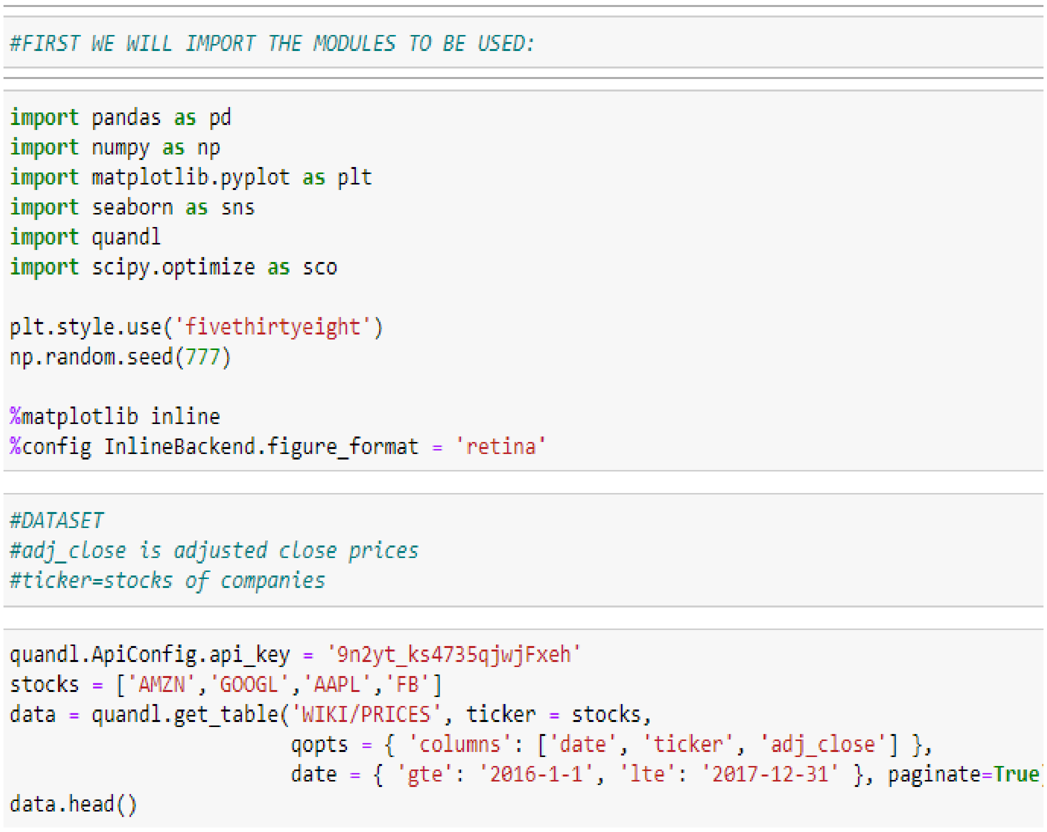
- We can see that with low-frequency data – large values of h – the ratio of asymptotic rates approaches the IID case.

- As h gets smaller – the higher the frequency – the relative performance of the variance estimator is better in the sense that the ratio of asymptotic rates gets smaller. That is, as the time dependence gets more pronounced, the rate of convergence of the mean estimator's MSE deteriorates more than that of the variance estimator.

---

2. CONSTRAINT EQUATION FOR OPTIMIZING WEIGHTS

- **ax+by=const given a+b=1 based on maximum return as well as minimum variance**



```
#FIRST WE WILL IMPORT THE MODULES TO BE USED:
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import quandl
import scipy.optimize as sco

plt.style.use('fivethirtyeight')
np.random.seed(777)

%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

```
#DATASET
#adj_close is adjusted close prices
#ticker=stocks of companies
```

```
quandl.ApiConfig.api_key = '9n2yt_ks4735qjwjFxeh'
stocks = ['AMZN','GOOGL','AAPL','FB']
data = quandl.get_table('WIKI/PRICES', ticker = stocks,
                        qopts = { 'columns': ['date', 'ticker', 'adj_close'] },
                        date = { 'gte': '2016-1-1', 'lte': '2017-12-31' }, paginate=True
data.head()
```

```
df = data.set_index('date')
table = df.pivot(columns='ticker')
# col[1] to select the stock names under multi-level column
table.columns = [col[1] for col in table.columns]
table.head()
```

	AAPL	AMZN	FB	GOOGL
date				
2016-01-04	101.783763	636.99	102.22	759.44
2016-01-05	99.233131	633.79	102.73	761.53
2016-01-06	97.291172	632.65	102.97	759.33
2016-01-07	93.185040	607.94	97.92	741.00
2016-01-08	93.677776	607.05	97.33	730.91

```
#GRAPH OF EACH STOCK
```

```
plt.figure(figsize=(12, 8))
for c in table.columns.values:
    plt.plot(table.index, table[c], lw=3,label=c)
plt.legend(loc='upper left', fontsize=12)
plt.ylabel('price in $')
plt.xlabel('Year')
```

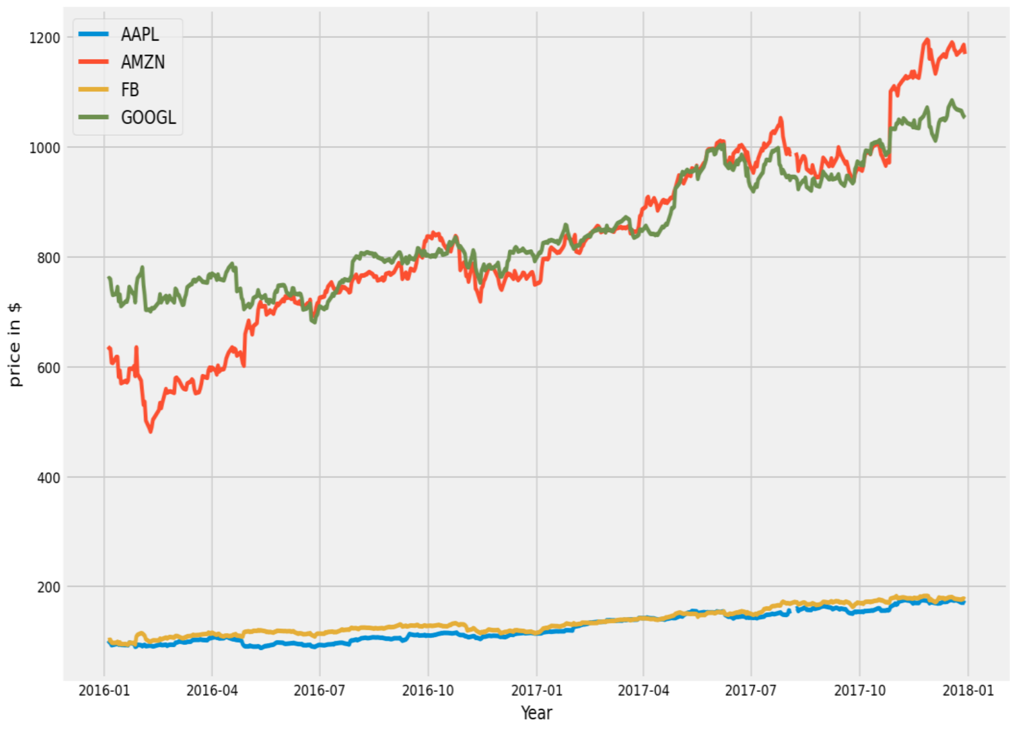


Fig: Graph of each stock.

```
table.rename(columns = {'AMZN':'x','GOOGL':'y'},inplace=True)
table.head()
```

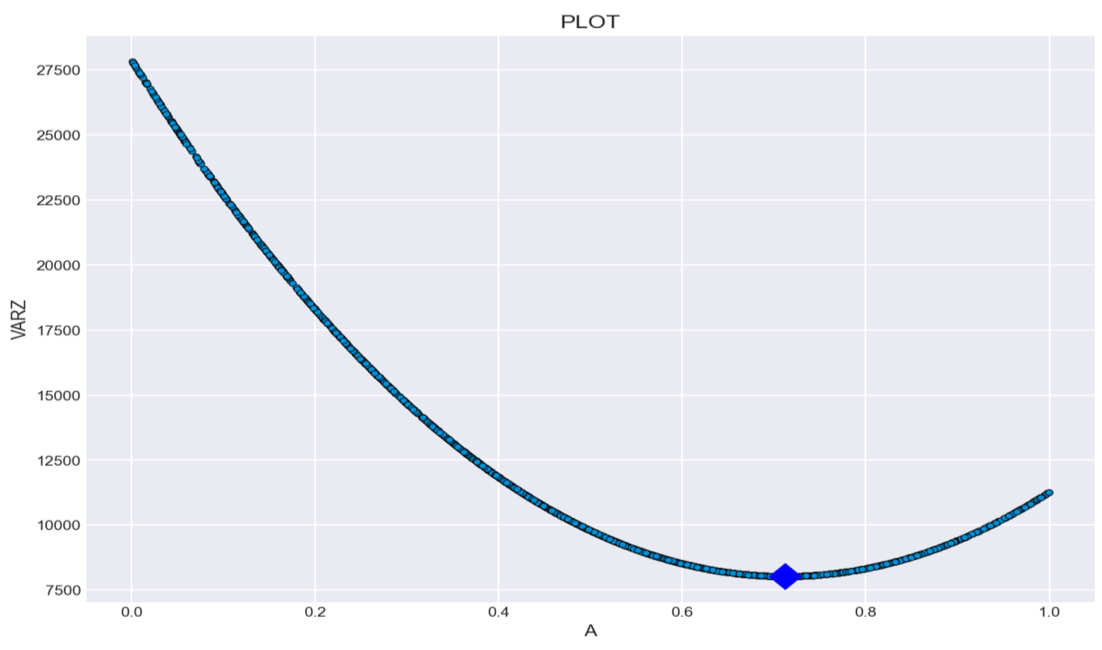
date	x	y
2016-01-04	636.99	759.44
2016-01-05	633.79	761.53
2016-01-06	632.65	759.33
2016-01-07	607.94	741.00
2016-01-08	607.05	730.91

```
xvar=table['x'].var()
yvar=table['y'].var()
xmn=table['x'].mean()
ymn=table['y'].mean()
returns1=table.pct_change()
returns1
covm=returns1.cov()
covxy=covm['x']['y']
print(xvar,yvar,xmn,ymn)
```

```
27881.472592713788 11274.020460802825 832.6615269461073 850.9057320717137
```

From here we can see that, variance of y is lesser than x, therefore a should have more weight. Now, let's see the result:

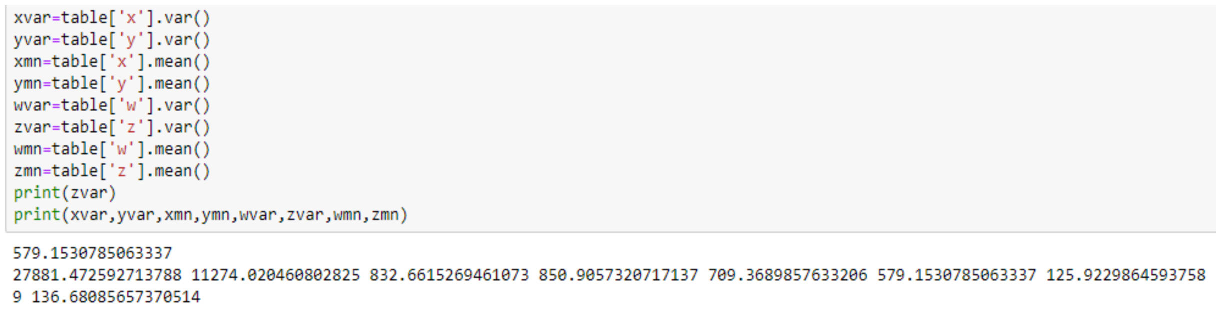
```
coeffs = []
zvar=[]
z=[]
a=[]
b=[]
num_assets = 2
num_portfolios = 2500
np.random.seed(101)

for single_portfolio in range(num_portfolios):
    coeff = np.random.random(num_assets)
    coeff /= np.sum(coeff)
    coeffs.append(coeff)
for i in range(num_portfolios):
    ai=coeffs[i][0]
    bi=coeffs[i][1]
    zi=(ai*xmn)+(bi*ymn)
    z.append(zi)
    zvari=(bi*bi*xvar)+(ai*ai*yvar)+(2*ai*bi*covxy)
    zvar.append(zvari)
    a.append(ai)
    b.append(bi)


    portfolio = {'Z': z,
              'VARZ': zvar,
             'A':a,'B':b}

df = pd.DataFrame(portfolio)

df.head()
```



```
Fig: Plot of variance vs weight(A).

Minimum Variance Portfolio
Z       837.916216
VARZ  8027.898060
A         0.711980
B         0.288020
Thus, Y should be given 71% and X 29%.
```

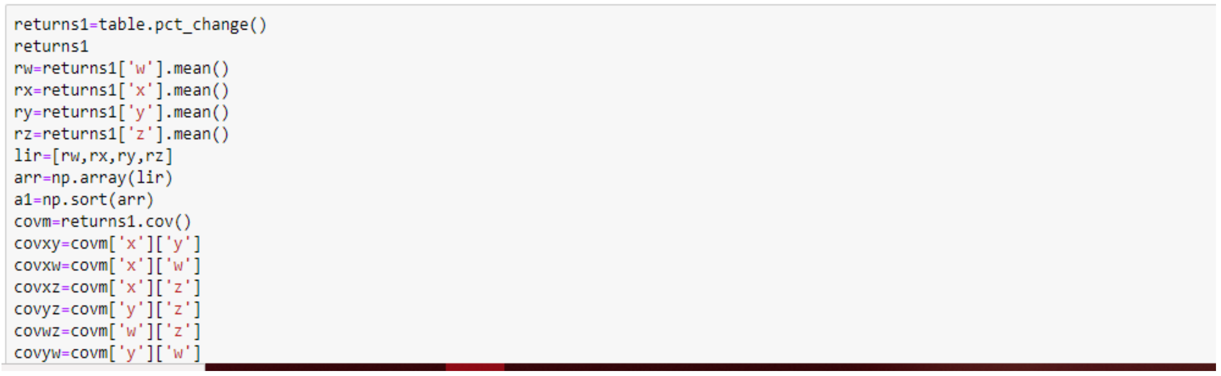
- **<u>ax+by+bz+bw =const given a+3b=1 based on maximum return as well as minimum variance</u>**

  In this part, we will give the maximum percentage to the stock with minimum variance and divide the rest amount equally in the other three stocks.



```
xvar=table['x'].var()
yvar=table['y'].var()
xmn=table['x'].mean()
ymn=table['y'].mean()
wvar=table['w'].var()
zvar=table['z'].var()
wmn=table['w'].mean()
zmn=table['z'].mean()
print(zvar)
print(xvar,yvar,xmn,ymn,wvar,zvar,wmn,zmn)
```

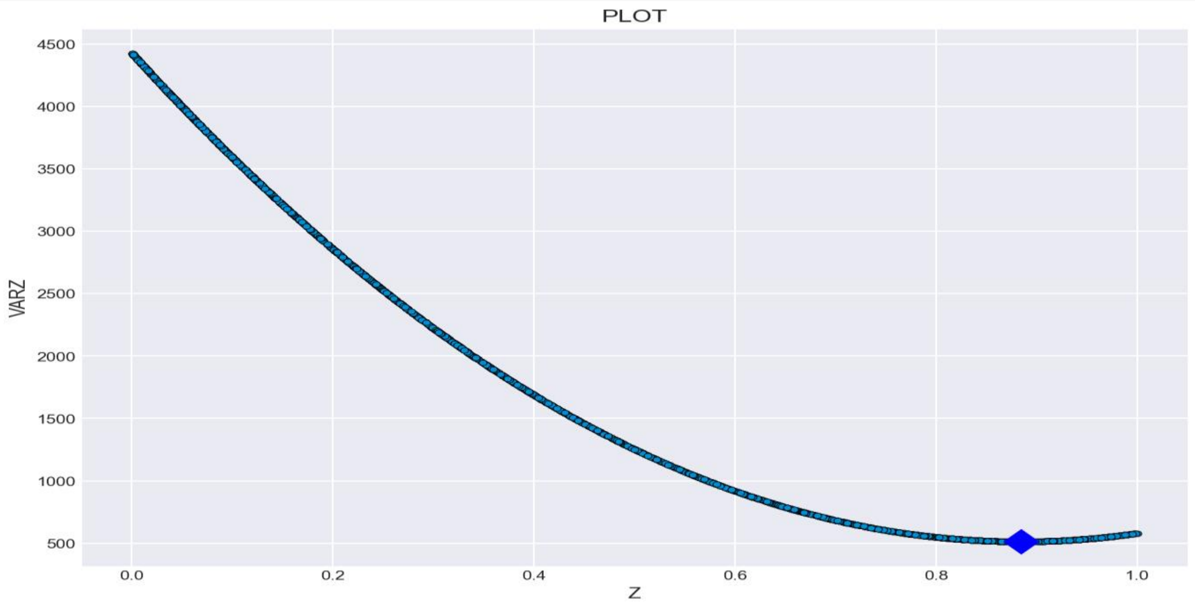
```
579.1530785063337
27881.472592713788 11274.020460802825 832.6615269461073 850.9057320717137 709.3689857633206 579.1530785063337 125.9229864593758
9 136.68085657370514
```



```
returns1=table.pct_change()
returns1
rw=returns1['w'].mean()
rx=returns1['x'].mean()
ry=returns1['y'].mean()
rz=returns1['z'].mean()
lir=[rw,rx,ry,rz]
arr=np.array(lir)
a1=np.sort(arr)
covm=returns1.cov()
covxy=covm['x']['y']
covxw=covm['x']['w']
covxz=covm['x']['z']
covyz=covm['y']['z']
covwz=covm['w']['z']
covyw=covm['y']['w']
```

Thus, variances of x,y,w,z are: 27881.472592713788 11274.020460802825 709.3689857633206 579.1530785063337
Therefore, we will give maximum percentage to Z as it have minimum variance and divide the rest to x,y and w.

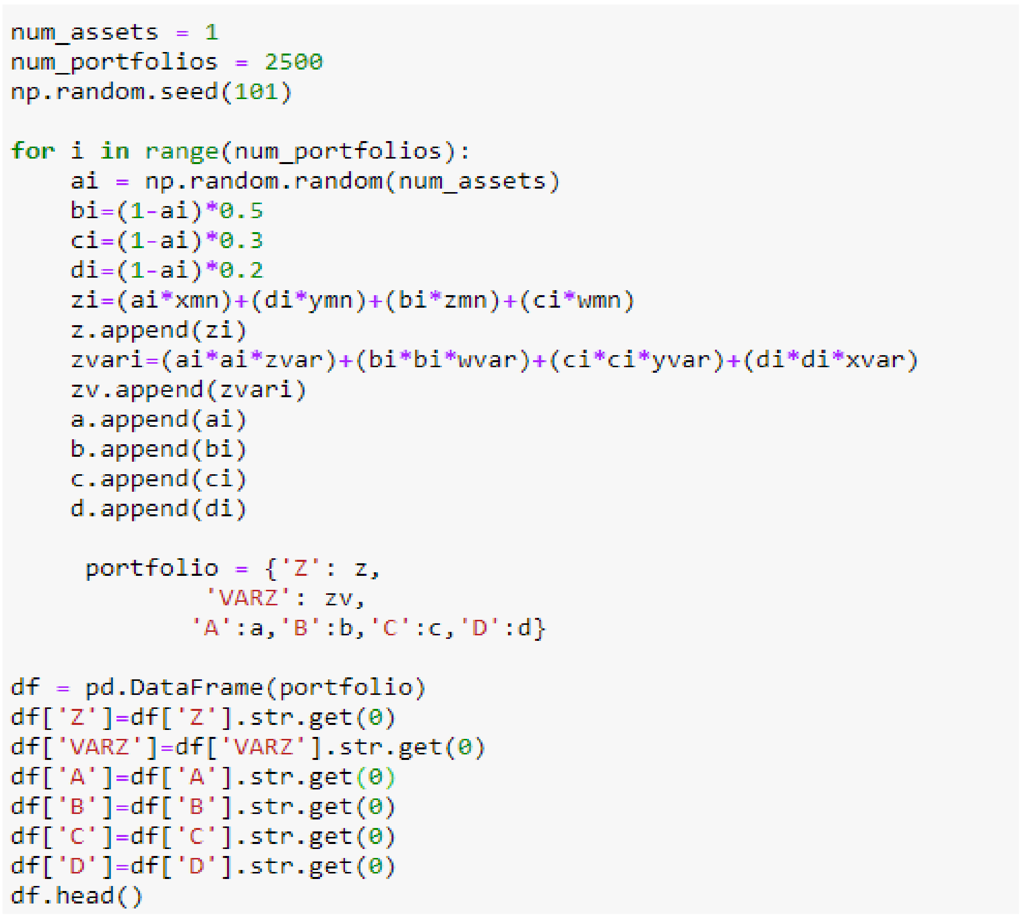
```
zv=[]
z=[]
a=[]
b=[]

num_assets = 1
num_portfolios = 2500
np.random.seed(101)

for i in range(num_portfolios):
    ai = np.random.random(num_assets)
    bi=(1-ai)/3
    zi=(bi*xmn)+(bi*ymn)+(ai*zmn)+(bi*wmn)
    z.append(zi)
    zvari=(ai*ai*zvar)+(bi*bi*yvar)+(bi*bi*xvar)+(bi*bi*wvar)+(ai*bi*(covyz+covxz+covwz))+(bi*bi*(covxy+covxw+covy
    zv.append(zvari)
    a.append(ai)
    b.append(bi)



    portfolio = {'Z': z,
             'VARZ': zv,
            'A':a,'B':b}

df = pd.DataFrame(portfolio)
df['Z']=df['Z'].str.get(0)
df['VARZ']=df['VARZ'].str.get(0)
df['A']=df['A'].str.get(0)
df['B']=df['B'].str.get(0)
df.head()
```



```
Minimum Variance Portfolio              1939
Z      190.535079
VARZ   512.184551
A        0.884553
B        0.038482
Hence, Z will get 88% and x,y,w 3% each.
```

- **ax+by+cz+dw=const given a+b+c+d=1 and b:c:d=0.5:0.3:0.2(1-a) based on maximum return as well as minimum variance**

As we already know, Thus, variances of x,y,w,z are: 27881.472592713788 11274.020460802825 709.3689857633206 579.1530785063337.
VarZ<VarW<VarY<VarX
Therefore, we will give maximum percentage to Z as it have minimum variance and from the the rest we will give 50 % to W, 30% to Y and 20% to X.

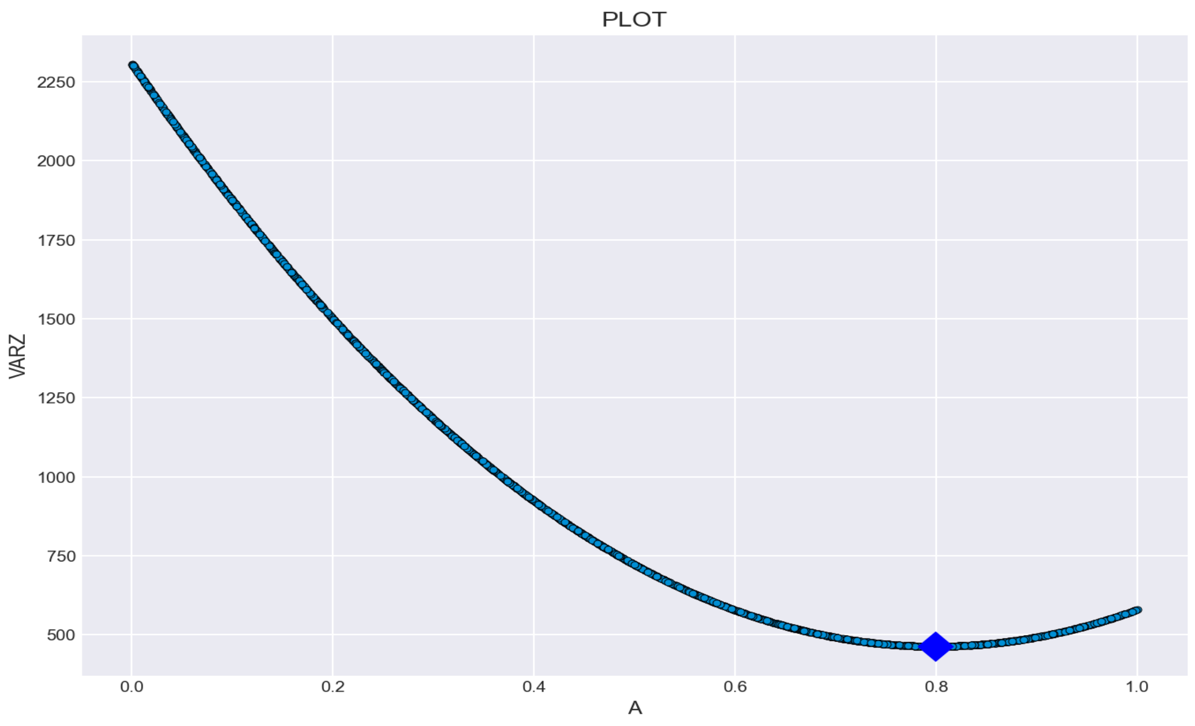


```
num_assets = 1
num_portfolios = 2500
np.random.seed(101)

for i in range(num_portfolios):
    ai = np.random.random(num_assets)
    bi=(1-ai)*0.5
    ci=(1-ai)*0.3
    di=(1-ai)*0.2
    zi=(ai*xmn)+(di*ymn)+(bi*zmn)+(ci*wmn)
    z.append(zi)
    zvari=(ai*ai*zvar)+(bi*bi*wvar)+(ci*ci*yvar)+(di*di*xvar)
    zv.append(zvari)
    a.append(ai)
    b.append(bi)
    c.append(ci)
    d.append(di)

     portfolio = {'Z': z,
             'VARZ': zv,
            'A':a,'B':b,'C':c,'D':d}

df = pd.DataFrame(portfolio)
df['Z']=df['Z'].str.get(0)
df['VARZ']=df['VARZ'].str.get(0)
df['A']=df['A'].str.get(0)
df['B']=df['B'].str.get(0)
df['C']=df['C'].str.get(0)
df['D']=df['D'].str.get(0)
df.head()
```

```
min_var = df['VARZ'].min()
min_var
min_VAR_port = df.loc[df['VARZ'] == min_var]
min_VAR_port
```

	Z	VARZ	A	B	C	D
662	721.146703	462.947409	0.799565	0.100218	0.060131	0.040087

```
min_var = df['VARZ'].min()
min_VAR_port = df.loc[df['VARZ'] == min_var]
plt.style.use('seaborn-dark')
df.plot.scatter(x='A', y='VARZ', edgecolors='black', figsize=(10, 8), grid=True)

plt.scatter(x=min_VAR_port['A'], y=min_VAR_port['VARZ'], c='blue', marker='D', s=200 )
plt.xlabel('A')
plt.ylabel('VARZ')
plt.title('PLOT')
plt.show()

print('Minimum Variance Portfolio',min_VAR_port.T)
```



```
Minimum Variance Portfolio               662
Z      721.146703
VARZ   462.947409
A        0.799565
B        0.100218
C        0.060131
D        0.040087
```

Hence, Z has got 80%, W got 10% , Y got 6% and X got 4% of total amount.

Chapter 4 – FUTURE SCOPE

Our research will study and apply the tools and techniques of Data Science for the optimization of stock prices to help investors invest in the way they want. The investors can divide their money in whatever ratio they decide and can get the optimized portfolio for that ratio. This will help in effective investment and profit making.

Both the models can be used by investors before investing in markets. Modern Portfolio Theory is already in use by many, but as we can see the Robust Portfolio Optimization is a better choice, so shifting of people from first model to later will also be done. Also, Robust Portfolio can be implemented using quantile statistics or by constructing near optimal portfolios or by hybrid heuristics algorithm.

Furthermore, just like the second model other models of optimization can also be implemented by differing the constraints of the equation. Different strategies can be used for optimization by differing skewness and sharpe ratio.

And also there is a possibility of doing performance analysis of models and comparing them to efficient frontier model. We can make a predictive model and an optimization model together for the best analysis of stock market and checking the accuracy of the model. Thus, for the investors with different requirements different models can be generated to give them the best experience.

Chapter 5 – CONCLUSION

This report details the process to develop the investment strategies for investors. Earlier, people used to invest in one asset a time but the introduction of Markowitz model brings the idea of diversification which proofs the increase in return when invested in different assets at a time as the return depends on the covariances of the asset. Later, it was found out that Markowitz portfolio optimization requires knowledge of the mean return vector and covariance matrix parameters. As it turns out, the resulting optimized portfolio is so highly sensitive to small estimation errors in such parameters that it is unusable in practice. One step towards the solution is to make the portfolio design robust to uncertainties in the parameters. The main contribution of the robust portfolio optimization approach lies in its robustness to heavy tails in high dimensions. Heavy tails present unique challenges in high dimensions compared to low dimensions. To design the robust counterpart of (6.1), here we assume that the uncertainty sets of the mean return μ and covariance matrix Σ are separable, convex, and compact . A conservative and practical investment approach is to optimize the worst-case objective over the uncertainty sets. Later we found out that as h gets smaller – the higher the frequency – the relative performance of the variance estimator is better in the sense that the ratio of asymptotic rates gets smaller. That is, as the time dependence gets more pronounced, the rate of convergence of the mean estimator's MSE deteriorates more than that of the variance estimator.

In the second part of this report, we have focused on the importance of variance in the optimization problems. We have taken different constraints and solved them to get the optimized weights for the investors by minimizing variance of the overall equation $z=ax+by+cz+dw$. We have assigned weights based on the order of the variance as well as return of each stocks. The investors can divide their money in whatever ratio they decide and can get the optimized portfolio for that ratio. This will help in effective investment and profit making.

RFERENCES

i. A signal processing perspective on financial engineering - Yiyong Feng .

ii. Signal Processing in Finance- https://sites.tufts.edu/eeseniordesignhandbook/2015/signal-processing-in-finance/

iii. Optimization of investment portfolio weight of stocks affected by market index- https://iopscience.iop.org/article/10.1088/1757-899X/166/1/012008/pdf

iv. Methods of Optimization in Finance- Gerard Cornuejols and Reha T¨ut¨unc¨ u

v. Udemy- Python for Data Science and Machine Learning Bootcamp.

vi. Analysis of portfolio optimization with lot of stocks amount constraint- Liem Chin1 , Erwinna Chendra2 , Agus Sukmana3- https://iopscience.iop.org/article/10.1088/1757-899X/300/1/012004/pdf

vii. Robust Portfolio Optimization- Fang Han, Huitong Qiu- https://papers.nips.cc/paper/5714-robust-portfolio-optimization.pdf

viii. https://blog.quantinsti.com/portfolio-management-strategy-python/#multiple-strategies

ix. Stochastic programming technique for portfolio optimization with minimax risk and bounded parameters P KUMAR1 , G PANDA2,* and U C GUPTA2

x. Michael J Best and Robert R Grauer. On the sensitivity of mean-variance-efficient portfolios to changes in asset means: some analytical and computational results. Review of Financial Studies, 4(2):315–342, 1991.