PORTFOLIO MANAGEMENT USING R SIMULATION AND ARMA STOCK RETURN PREDICTION

Capstone Design
March 2017
A. Salhi
Supervised by N. Rhiati
Capstone Student: Ali Salhi

Approved by the Supervisor

SUPERVISOR: Nabil Rhiati
Acknowledgments page

This project would not have been possible without the help and support of many individuals. First I would like to thank my supervisor who has helped me throughout this semester mainly by setting deadlines for me and helping me respect them to the best of his ability as well as by looking at my project through the eyes of a demanding customer seeking perfection. I would also like to thank Mr. Tahar Harkat who provided me with the financial expertise and guided me through a big part of this project by telling me which calculations to make and by showing me which results were relevant and which were not.

I am also grateful to anybody that I have interacted with and talk to about my project and that has provided me with great feedback. A lot of times people who aren’t related to any of the areas of expertise that are included in this project gave me the best feedback and helped me see things from their perspectives.
### Contents

1. Abstract 6
2. Introduction 7
3. Methodology 9
4. Financial concepts 11
   4.1 Portfolio Management 11
   4.2 Efficient Frontier 12
   4.3 Portfolio Covariance Matrix 14
5. Mathematical Concepts 15
   5.1 Mathematical Modelling 15
   5.2 The Data and Model in use 16
6. Technology Enablers 21
   6.1 The “R” Programming Language 21
   6.2 Why “R” 21
   6.3 R Studio 22
   6.4 Packages 22
   6.5 Packages Used 23
      6.5.1 Quantmod 23
         6.5.1.1 GetSymbols 23
      6.5.2 Portfolio Analytics 24
         6.5.2.1 Portfolio.Spec 24
         6.5.2.2 Add.Constraint 24
         6.5.2.3 Random_Portfolios 25
         6.5.2.4 Add.Objective 25
         6.5.2.5 Optimize.Portfolio 25
      6.5.3 Plotly 26
      6.5.4 Forecast 27
         6.5.4.1 ARIMA 27
         6.5.4.2 Forecast 27
      6.5.5 Stats 28
         6.5.5.1 NA.omit 28
         6.5.5.2 Predict 28
7. The Program sequence 29
8. The output file 31
9. Conclusion 33
10. References 34
11. Appendix 1 36
12. Appendix 2 49
# Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Example of a stationary time series</td>
<td>18</td>
</tr>
<tr>
<td>2</td>
<td>Example of a non-stationary time series</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Adjusted closing price of INFOSYS over 1 year</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>The R Studio interface</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>Opening the R script on R studio</td>
<td>37</td>
</tr>
<tr>
<td>6</td>
<td>Loading the libraries to the current session</td>
<td>38</td>
</tr>
<tr>
<td>7</td>
<td>Prompt for the number of assets to invest in</td>
<td>38</td>
</tr>
<tr>
<td>8</td>
<td>Prompt for the start and end date for data retrieval</td>
<td>39</td>
</tr>
<tr>
<td>9</td>
<td>Entering 5 stocks and their index</td>
<td>39</td>
</tr>
<tr>
<td>10</td>
<td>IBM OHCL data frame</td>
<td>40</td>
</tr>
<tr>
<td>11</td>
<td>Prompt for constraints to be added to the portfolio</td>
<td>40</td>
</tr>
<tr>
<td>12</td>
<td>Suggestions and warning messages from the program</td>
<td>41</td>
</tr>
<tr>
<td>13</td>
<td>Script to visualize the portfolios plot with Risk VS Return</td>
<td>41</td>
</tr>
<tr>
<td>14</td>
<td>Random portfolios plot</td>
<td>42</td>
</tr>
<tr>
<td>15</td>
<td>Script to visualize efficient frontier</td>
<td>42</td>
</tr>
<tr>
<td>16</td>
<td>Efficient frontier data frame</td>
<td>43</td>
</tr>
<tr>
<td>17</td>
<td>Weights of assets making up the efficient frontier</td>
<td>43</td>
</tr>
<tr>
<td>18</td>
<td>Data frame showing the weights of assets within specific portfolios</td>
<td>43</td>
</tr>
<tr>
<td>19</td>
<td>Script to retrieve weights of assets making up a specific portfolio</td>
<td>44</td>
</tr>
<tr>
<td>20</td>
<td>Results from single portfolio weights retrieval</td>
<td>44</td>
</tr>
<tr>
<td>21</td>
<td>Script for retrieving manually computed values for individual stocks</td>
<td>45</td>
</tr>
<tr>
<td>22</td>
<td>Values for the IBM stock</td>
<td>45</td>
</tr>
<tr>
<td>23</td>
<td>Script for retrieving the automatically generated summary for each stock</td>
<td>46</td>
</tr>
<tr>
<td>24</td>
<td>Summary for the IBM stock</td>
<td>46</td>
</tr>
<tr>
<td>25</td>
<td>Script to plot the historical return and the predicted ones</td>
<td>47</td>
</tr>
<tr>
<td>26</td>
<td>Plot of the historical returns and predicted ones</td>
<td>48</td>
</tr>
<tr>
<td>27</td>
<td>Accuracy measures of predicted VS actual returns</td>
<td>49</td>
</tr>
</tbody>
</table>
1. Abstract

This report covers all the aspects of my capstone project. It is structured in such a way that a person can understand all of the work that was done with no prior knowledge about the fields of finance, computer science, and mathematics. The introduction talks about all the fields that are involved in the making of this project with a preview of the workings and intent behind the work performed, which is to help portfolio managers make better informed decisions using analytics tools as well as a mathematical model. After that, and since this is a software project, a methodology section introduces the software engineering methodology that was used to keep all of the stakeholders involved throughout the development phase. A well detailed description of the technology enablers used follows that. Towards the end there is a full description of the whole code from the beginning to the end followed by a demo with screenshots and snippets of code.
2. Introduction

Data science is a fast-growing field where various methods and processes are applied to raw and usually large sets of data to extract valuable information. Data science is not a discipline on its own but rather an interdisciplinary field that merges techniques from mathematics, statistics, and computer science. This large area of studies is further divided into four more detailed and specialized subfields, namely: Decision Analytics, Descriptive Analytics, Predictive Analytics, and Prescriptive Analytics. The two that will be used extensively in the project are prescriptive as well as predictive analytics. The first will be used to make a recommendation to the user on what their best option of investment is. The recommendation will be based on a simulation to which we will apply optimization techniques. The second one will be used to predict future returns of certain stocks. That will help the portfolio manager make financial projections and run different scenarios. Now following this logic and using tools and techniques from prescriptive as well as predictive analytics, the goal of this project is to provide portfolio managers with a tool they can use to get insight from structured historical data about assets, mainly stocks, to make better informed decisions when handling their client’s money. [1]

To be able to perform simulations and use historical stock prices to make decent decisions on where to invest your money, you should have an extensive knowledge in the financial field as well as some training on a statistical software. The problem is that these two prerequisites are not within everybody’s reach, and they themselves require time, energy and money. The only other two alternatives are to invest your money at your own risk without any insight or prior knowledge of the market, or hire somebody to invest your money and take a cut.
Now with that being said, this program’s main purpose is to be able, through an interactive interface, to give some sort of guidance to people that want to create a portfolio of assets to invest in, maximize their returns, and make some money in the process. The user doesn’t necessarily need to be an expert in the fields of computer science or finance to be able to use it.

My knowledge about the fields of data science and finance at the beginning of the project was trivial. I was exposed to some of the technologies used in analytics by pure chance while researching it online, as in for finance, I learned about it in one class I took while studying abroad. I took this project as an opportunity to learn and absorb as much knowledge as possible about these two fields because Business and Financial Services Analytics are fields that I am highly interested in for my future career.

This report will contain all the information that relates to the project whether that be detailed technical information or methodologies used and organization followed to conduct the whole project and take it from the design stages to the concrete realization and testing. The report will be complemented by screenshots of its workings to better illustrate the descriptions given and provide the reader with a visual idea of the final product.
3. Methodology

The main goal behind this capstone project being the development of a computer program I relayed heavily on the methodologies I have learned in my two software engineering classes. These methodologies are there to ensure smooth execution of the project and a good flow throughout the progress towards the final and desired result. After a review of the many methodologies available I judged the best fit for the size and nature of this project to be the “Agile Scrum Methodology”.

Agile methods were created due to the continuous problems caused by the huge amounts of overhead involved in the design of new software. Agile methodologies favor small steps and incremental development with the minimal necessary planning over long-term unnecessary planning that might be subject to change because of the nature of the fast-paced environment that we live in. Following these methodologies guarantees frequent delivery of new functionalities that are added with every new increment. Agile also favors face to face communication over written documents to avoid any ambiguity and misunderstandings. The main principles of Agile methodologies are best illustrated through the Agile manifesto:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

(Agile manifesto)[2]
Agile methodologies have their limitations as well. Some of the ones that were faced during the development of this project are the difficulty to keep all the stakeholders interested in the project as well as maintaining simplicity. The first one is because most of the stakeholders don’t have a background in computer science and get bored fast when programing technicalities and difficulties are discussed. The second one is because there is a constant need to cope with change and that entails adding or deleting parts of the program which doesn’t help keep it simple and straightforward. [3]

The Agile Scrum methodology is the closest to what was adopted during the development of this process. The term closest is used because this project was developed with only one programmer and not a team. The main goal of the scrum is to provide a process that favors iterative development and continuous customer involvement. The first phase of the Scrum is where the general outline and objectives of the project are determined with the participation of the programmer and the customer. This phase is what gave birth to the project specification document. The second phase is the longest since it’s the one of the development which happen in small increments. These are called sprint cycles with each sprint resulting in an increment to the program. The third and final phase of the scrum is the one that completes the required documentation, does an assessment of the project and performance of the time and completes anything that is left other than code. [3]

The sprint being the main component of the Scrum process, it is divided into four distinct phases that are followed daily to make sure that everything is in order. These phases are:
- Sprint planning meeting: A collaborative meeting between the customer and the team to determine what the next step in the development process is. (example: plotting the efficient frontier)

- Sprint: The increment to the program is designed, coded and tested.

- Daily scrum: A daily quick meeting where questions like “what has been done?”, “what should be done?”, and “What obstacles are in the way?” are answered.

- Sprint review meeting: Customer discusses what has been done and the team showcases the work with a small demo, after which they talk about the problems they faced and the ones they fixed. [3]

4. Financial concepts

Before getting to the program and what it does in details, you need to have a certain background in finance. The goal of this section is to introduce the necessary financial concepts and terminology that will be used in future sections. I will start by explaining what a financial portfolio is and follow by introducing the concept of portfolio management and the efficient frontier theory.

4.1. Portfolio management

A portfolio is a combination of a multitude of financial assets such as stocks. Portfolios are owned by investors and managed either by them or by trained financial professionals called portfolio managers. A portfolio then is an unequal distribution of one’s money into different assets with the goal of making a positive return and creating wealth.
Portfolio management is the science of taking somebody else’s money and making the right choices to invest it in different assets to meet the investor’s goals and what they have in mind. It’s very important for a portfolio manager to know why a person is investing their money. For instance a portfolio manager dealing with a person saving to afford tuition fees for his/her children in the coming 3 years will make different investment moves than one dealing with a person saving for retirement in 15 years. This means that each portfolio differs from another in terms of the constraints it is subject to. These constraints are defined by both the investor and the portfolio managers and can be written in terms of liquidity (for example if the investor will withdraw money in a period of one or 2 years, the portfolio manager will make investment in liquid investment, or investment that can easily be converted into cash.), in terms of maximum or minimum risk, in terms of minimum or maximum return, in terms of diversification (or distribution of risk), and so on. [4]

The most important task of a portfolio manager is to be able to determine the perfect mix of assets to invest in to meet the customer’s needs by investing is a well-diversified portfolio that meets the required return on investment expected by the customer with the minimum risk possible. [4]

4.2. Efficient Frontier

The efficient frontier represents the different combinations of weights of a given portfolio of assets. These set of portfolios either have the highest return for a predefined risk or on the opposite the lowest risk for a predefined return. The efficient frontier concept is the work of Harry Markowitz, a Nobel Prize laureate. This concept makes a lot of assumptions that form a
relatively representative image of the reality of the market. Some of these assumptions are that all investors are risk averse, more risk automatically means a higher return, and that asset returns follow a normal distribution. [4]

Investors being risk averse means that when presented with multiple options that have the same expected return with varying degrees of risk they will choose the one with the lowest risk. More risk means more return is self-explanatory, it means that the higher the risk you are willing to take the higher the reward you may receive. The third assumption means entails that asset returns will always stay within three times their standard deviation. This latter is usually proven wrong by the reality of the market. [4]

In the same fashion the modern portfolio theory was developed by Markowitz and it stipulates that investors can create portfolios with an optimized return for a certain risk. This theory comes in the form of a process containing four phases that are:

- Security valuation
- Asset allocation
- Portfolio optimization
- Performance measurement

In order to construct an efficient frontier, some variables need to be computed. For this, each one imported set of historical prices for a given stock requires a computation of the standard deviation that is a unit of risk, and the average return of its daily returns.

The standard deviation is given by the formula:
\[ \sigma = \sqrt{\frac{\sum (X_i - \bar{X})^2}{n - 1}} \]

Where:

\(X_i\): are the daily returns.

\(\bar{X}\): is the average return.

\(N\): is the number of periods.

While the formula of the arithmetic return is given as:

\[ r = \frac{\sum \frac{P_t - P_{t-1}}{P_{t-1}}}{n} \]

Where:

\(P_t\): is the price at day \(t\).

\(P_{t-1}\): is the price at day \(t-1\).

\(N\): is the number of periods.

4.3. Portfolio Covariance matrix

The covariance matrix for a given portfolio is one that displays the covariance between each two assets together. This indicator will give the portfolio manager some insight about how two assets are specifically related. This insight is used to determine if two assets are positively correlated and move together (If one goes up the other one does too), or if they are negatively correlated and move on opposite directions with respect to each other (If one is going up the other one is going down) [5]. The formula to compute the covariance between two stocks is the following:

\[ \text{Covariance} = \frac{\sum (\text{Return}_{ABC} - \text{Average}_{ABC}) \times (\text{Return}_{XYZ} - \text{Average}_{XYZ})}{(\text{Sample Size}) - 1} \]
Where:

Return$_{ABC}$: Is the return one stock ABC

Average$_{ABC}$: Is the average return on ABC

Return$_{XYZ}$: Is the return one stock XYZ

Average$_{XYZ}$: Is the average return on XYZ

5. Mathematical Concepts

The second part of this program relays heavily on mathematical concepts to give a prediction about future stock returns over a 25-day period. Therefore, some mathematical concepts need to be discussed to put it in perspective and have everything clear. I will first explain what a mathematical model is, talk about the model I will be using, and then talk about all the assumptions made and how to make sure they are respected.

5.1. Mathematical modeling

A mathematical model is a representation of a certain system using mathematical formulas and concepts. In our case the system to be represented is the movement and fluctuations of returns in the stock market. Mathematical models are used to study the building blocks of a certain system and their past behaviors with a possibility to find trends and patterns to predict future behavior. Mathematical models contain multiple types of information, some of the most important ones are variables and relationships. Because of the nature of the mathematical language relationships are represented by logical operators and the variables are usually the terms describing our system.
The multitude and complexity of mathematical models gave birth to classification rules that are used to differentiate between certain types of models. The first concept is the one of linearity. This latter means that a relationship between two elements can be graphed as a straight line. When a model is non-linear it is described as chaotic or random, which in our case is inadmissible because we want to use this model to predict future returns with a certain degree of accuracy. The second differentiation is static modeling versus dynamic modeling. A dynamic model is one that takes into consideration the time variable. That means that whatever system we are trying to model evolves over time. In our case, returns fluctuate daily so our model must be dynamic. The third important characteristic by which we will describe our model is whether it is deterministic or probabilistic. A probabilistic model is one where the state of a certain variable (in our case: returns) is described not by certain values but by a probability distribution of its many states over time. Based on that we can describe our model as probabilistic and not deterministic [6].

5.2. The Data and model in use

The input data to the model in use is going to be historical values of log returns of a given stock. Our data then fits exactly the definition of a stochastic process. A stochastic process is one who’s evolution over time follows a set of probabilistic laws. Such processes are made up from a set of random variables. Based on that the model that will be used for the purpose of this project is the Autoregressive Moving Average model.

ARMA (Autoregressive Moving Average) model is known to be have decent performance when it comes to predicting short terms stock returns. The ARMA model is one that generates a
description for stationary time-series based on two polynomials (more on stationary time-series later). The first is used to model the auto-regression and the second to model the moving average. If we take the time-series describing the historical stock prices of a certain company for example, the ARMA model will be used to understand the tendencies and behavior of the series, and to predict possible future values. The AR part will focus on regressing the variable (daily returns) on its own historical values. After that the MA part will model the standard deviation (error) as a linear combination of standard deviations (errors) happening contemporaneously and at various times in the past [7].

This model is denoted as ARMA(p, q) with p and q being parameters entered by the user of the model (that would be me in this project). Since this model is a combination of two sub-models, each of the two variables is aimed at one part of it. The first one “p” is the order of the autoregressive model, and “q” is the order of the moving average model [7].

The equation for the autoregressive part of the model is as follows:

\[ X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t. \]

Here we can see that \( X_t \) is derived by linearly regressing terms from \( X_{t-1} \) up to \( X_{t-p} \) which explains the name autoregressive. On the other hand the equation for the moving average part of the model is as follows:

\[ X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}. \]
The ARMA model is a combination of the two and is denoted as follows:

\[ X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}. \]

One assumption with the ARMA model is that it takes in stationary time series. A stationary process is one whose joint probability distribution doesn’t change when shifted in time. What that entails is that values such as the mean and the variance of the series will stay consistent over time[8].

An example of a stationary process is something like this following graph showing a white noise series.

Figure 1: example of a stationary time series
A non-stationary process looks like the graph below showing deviations in the global temperatures over a period of time:

Figure 2: Example of a non-stationary time series

Now with both of those graphs above we can look at the plot of any given stock throughout time and figure out whether stock prices are stationary or not. Below is the plot for INFOSYS’s adjusted closing price over a 1 year period:
Judging from the graph above, we can clearly see that stock prices not a stationary process. Knowing that ARMA is only valid on stationary time series we need to change it to a stationary one. There are many ways to make non-stationary times series into stationary ones. One way to do that is to use differencing, which consists of computing the difference between two adjacent observations. R’s base package contains a function called “diff” that returns lagged and iterated differences (The function will be discussed later on). Details on how this model is used and integrated into the program are discussed later on in the report [8].
6. Technology Enablers

6.1. The R programming language

R is a programming language mainly used for computation intensive tasks. Usually referred to as a statistical programming language R is extensively used by statisticians, data scientist and data mining professionals because of the way it handles data sets and computations on different arrangements of data such as matrices. R is available on the “r-project” website for free and under the GNU public license. That means that full comprehensive documentation is provided by the R-project team and that the software is free to use, modify and redistribute by anybody [9].

6.2. Why R

One crucial part that comes at the early stages of every software development project is the moment where the team (or individual developer) decide on the technology enablers that best fit their needs. These days the market leaves us no short of statistical software to choose from. There is a wide variety of commercial and open source options to choose from. R’s appealing characteristics start with the fact that it’s a programming language and offers almost infinite extensibility through its many packages. Before talking about its packages, R’s base functions and its design are also appealing and fit the nature of this project very well. Some of the main advantages of R are its very effective storage capacities, its various operations on vectors and matrices as well as a wide range of graphing facilities that make the presentation of results effective and convenient [9].
6.3. **R Studio**

There are many ways to interact with R, when you download the language it automatically comes with a command line prompt where you can write scripts and start using it. The needs of this project being software development and not just script writing, a decision was made to use an integrated development environment. Although there is a wide range of IDEs to choose from, I decided to use R studio. This choice was made because of the functionalities of this IDE as well as its graphical user interface ("Rstudio"). Once the IDE is launched it presents users with a screen divided into four parts [10]. These parts are arranged as follows:

- The top left corner contains the R script files with extension ".R".
- The bottom left corner contains the command line prompt where the user can enter commands (individual or multiple R scripts to test them).
- The top right corner contains the Data environment where all the open variables and data sets that are working at a certain point are available for access and view.
- The bottom right corner contains a multi-tab section with the current directory, the plots, the packages available, a help section and a viewer (web results) [10]

6.4. **Packages**

One of R’s main features and attractive characteristics is its extensibility. The Comprehensive R Archive Network (CRAN) contains as of March 2017, an outstanding 10197 packages. Most of those packages are written by individuals and are available for download for free directly from the command line using the script:

`>> Install.packages("Name of Package")`
Once this command typed R takes care of locating the package on its archive network and download the necessary binaries and executable files for it. After that is done all that left for the user is to include the appropriate library and start enjoying new functions [11].

6.5. Packages used

6.5.1. Quantmod

Quantmod or the Quantitative Financial Modeling Framework is a package available in the comprehensive R archive network and provides its users with functions and tools to retrieve data from the internet, and build trading models that are based on statistical modeling and testing. Quantmod is also described as a rapid prototyping environment, that this package is designed to provide straightforward, easy to use function that will deliver the required functionality without a large learning curve [12].

6.5.1.1. getSymbols

This function is used to retrieve OHCL objects from a variety of sources. These sources can either be local like comma separated variables files that are stored on your computer or remote sources like Yahoo Finance. OHCL objects are usually historical stock prices with the Opening, High, Closing, Low, Volume traded, and Adjusted values during a specific period [12].

This function can take many arguments that offer some flexibility and manipulation of how the function behaves. The most used ones in this project are “from.date” and “to.date” that mark the beginning and end of from where to start and finish retrieving the data. If these two arguments are not specified, the function retrieves all the available data from the specified source [12].
6.5.2. Portfolio Analytics

The second symbol, Portfolio Analytics is perhaps the most important technology enabler in this whole project. This package contains a pool of functions and a comprehensive documentation that are vital to any portfolio manager. It provides portfolio managers with numerical solutions and the ability to set return objectives and constraints on a set of portfolios. The main optimization methods that are currently presented by this package are random portfolios, differential evolution, particle swarm optimization, generalized simulated annealing, linear and quadratic programming routines. These methods can be coupled with a multitude of solvers that are provided by other packages. The solver that will be used in this specific project is part of R’s optimization infrastructure (ROI) that will be discussed further down [13].

6.5.2.1. portfolio.spec

This function is used to create a portfolio that will eventually hold information like the constraints and the weights. This function takes as argument the assets that will be contained in the portfolio. In our case, we will give this function a vector (asset) that holds the user’s input on what stocks they would like to include [13].

6.5.2.2. add.constraint

This function always comes after creating a portfolio with the previous function. It either adds constraints to our portfolio or updates existing ones. There are several constraints that we can choose from, some of the most important ones are:

- Weight constraints: these can specify what the sum of all the weights must be or what is the minimum or maximum weights that can be applied to a certain asset.
-Diversification constraints to ensure that the result of our simulations will always give us a well-diversified portfolio.

-Return constraints to specify what the expected return must be for the portfolio to be acceptable [13].

### 6.5.2.3. random_portfolios

This function has two definitions in the Documentation for the Portfolio Analytics package. The way this function is used in the program follows the second definition. This function takes three arguments, the first one being the portfolio that needs to be randomized, the second one specifies the number of permutations that need to be made of that portfolio and the third one specifies the method that should be followed to do the randomization [13].

### 6.5.2.4. add.objective

This function’s purpose is to align the business objectives of the portfolio manager with how the program behaves. The way it’s used in the program is with three arguments as follows:

- An object of type portfolio.spec that we created earlier.

- Type of the objective that can either be a return, a risk, a return-budget, a quadratic-utility, or a weight concentration.

- Name of the objective can either be the mean, the median or other [13].
6.5.2.5. optimize.portfolio

This function provides a wrapper of all the functions that we described previously in this package to send all the information including the portfolio with the constraints, the expected performance and the set of weights that were randomized previously to a solver to perform optimization. For this project, our function is going to take four arguments as follows:

- The first one is an object of type xts or zoo or other R data type that contains assets returns.
- The object where we stored the add.objective function to provide our optimizer with objectives.
- The optimization method to be used, in our case the solver provided by R’s optimization interface.
- The random portfolio weights that were previously generated with the function random.portfolios, this will be the set that the optimizer works on [13].

6.5.3. Plotly Package

Plotly is a web based visualization tool mainly used for analytics. It is a commercial product used in many areas that require advanced graphing utilities and interactive, web-based plots. In this project the plotting of the efficient frontier was taken from the “Plot.ly” website. Plotly offers a package that is part of the CRAN. This package is free for download and use by anybody. The plot contains different colors with risk VS return significances. It seemed like the best choice for the purposes of this program because of its flexibility as well as the extensive community support available for it [14].
6.5.4. Forecast package

The title of this package is “Forecasting Functions for Time Series and Linear Models”. It contains a set of tools and functions used for the analysis and modeling of time series. The methods that are included in the package provide summaries for the time series we wish to work for. These summaries are used to understand the behaviors and workings of the series. The package also includes forecasting methods used to do projections and estimations of future values of these series [15].

6.5.4.1. ARIMA

In this program we will use this function to fit the ARMA model to our data. The use of this function comes from the ease of its integration with the predict() and forecast() methods that will be used later on. The ARIMA function is the exact same thing as the ARMA function but with an additional term that is the order of integration that we can modify. For the purpose of this project and because we want to use a simple ARMA model we will keep that coefficient null [15].

6.5.4.2. Forecast

This is the most important function in the second part of this project. It’s a generic function for forecasting time-series models. The function is a wrapper and invokes a specific detailed function depending on what model is used. In our case forecast() will invoke forecast.arima() because of the model we fitted previously [15].
6.5.5. Stats package

6.5.5.1. Na.omit

The stats package provides a multitude of functions to handle “NA” values. NA stands for “Not Available” and those are values that couldn’t be computed due to constraint problems or mathematical incompatibility and other various reasons. In our program some functions and calculations on vectors and matrices produce “NAs”. One example of that can be the computation of log returns of the closing stock prices of a certain asset. Although this is acceptable from a logical point of view, some of the functions that will use the resulting data don’t handle “NA” values and for that reason we need to get rid of them before going to the next steps. NA.omit is a very simple but vital function for the functioning of this program [16].

6.5.5.2. Predict

This is a generic function used for predictions based on the fittings of various models. These models have to be specified and fitted before the use of the “predict function”. The function adapts to whatever the class of its first argument is. In our case the first argument takes in the ARMA model we fitted before. The Predict function returns an object of type “forecast” this object contains a lot of important information in regards to our prediction of future returns [16]. Each and every “forecast” type object will contain the following:

- The model that the time-series was fitted to
- The name of the method that was used in the forecasting
- The point forecasts as a time-series (following the data that was used to train the model)
- A lower and upper boundaries of where the prediction lives
- The level of confidence that the return will live between the boundaries
- The time series that was used to train the model
- The residual values (time series minus the fitted values)
- The fitted values

7. The program sequence

- Get the number of assets to work with and store it in a variable "assets"
- Create the vector of size "assets" to hold the name of the stocks
- Get a start and an end day for the stock data to be retrieved
- Get the names of the assets from the user and store them in the vector "asset"
- Retrieve the OHCL data from the yahoo finance using quantmod
- Get the index from the user and then retrieve its OHCL from yahoo finance
- Get rid of the '^' sign from the index name because it's a special character
- Get the closing price of the index and store it in a data frame
- Compute the daily returns for the index and store them in a data frame
- Loop to compute the Geometrical mean of the daily returns of the index
- Store the arithmetic mean, standard deviation, geometrical mean and the beta value of the index
- Loop for each asset and do the following:
  - Get the closing price and store it in x
  - Compute the daily return and store it in dailyR
  - Compute the geometrical mean
  - Store the arithmetic mean and standard deviation and geometrical mean and beta value of each asset
- Save a mean of the returns of each asset in meanReturns
- Create a portfolio using “portfolio.spec”
- Add constraints to our portfolio using “add.constraint”
- Generate random portfolios using “random_portfolios”
- Get minimum variance portfolio using “add.objective” of type risk
- Optimize based on the minimum variance portfolio using the “optimize.portfolio” function
- Generate a maximum return portfolio using “add.objective” of type return
- Optimize based on the maximum return portfolio using the “optimize.portfolio” function
- Generate a vector of returns that could hold 100 values
- Build an empty efficient frontier data-frame with a column for risk and return
- Create a vector or matrix of to hold the weights of each asset for each one of the portfolios of the efficient frontier
- For each of the 100 spots available on the efficient frontier data frame do the following:
  - Create a portfolio and add to it the constraints that were specified before
  - Optimize that portfolio based on the constraints specified and using the ROI optimizer discussed earlier
  - Compute the risk
  - Compute the return for
  - Compute the sharp ration
  - Store all these values in the data frame
  - Get the corresponding weights that make up that portfolio
- Plot the resulting efficient frontier with the Plotly package and visualization tool
-For each of the assets entered by the user do the following [17]:

  - Get the log returns of the closing prices and make them stationary using the `diff()` function
  - Store the stationary data in a variable (time series)
  - Delete the NA values for the returns
  - Take 90% of that data and store it in a training time series
  - Take the remaining 10% and store them in a testing time series
  - Fit the training set to an ARMA model using the ARIMA function with the order of integration equal to zero
  - Use the stats predict function to get some projections in the future
  - Use the forecast function to project over 25 periods and get a more detailed object containing most of the important information about the model in use and the results

**8. The output file**

This program is intended for portfolio managers. One assumption that is made is that these portfolio managers, while proficient in finance, are not acquainted with tools such as R. With that assumption in mind, the results of the simulation and computations made by the program have to be easily accessible by the intended users. The solution to this problem is the creation of an output file with very detailed comments showing the user what scripts to run in order to retrieve a certain information.

The output file contains a multitude of scripts showing different results to the user. This file doesn’t include information retrieval scripts for intermediary results that are only computed as
part of a bigger formula. The results shown were the ones judged important from a portfolio manager’s point of view. Results that can provide the user with insight and affect their decision. These are:

- A script for the visualization of the portfolios selected and computed on an interactive graph.
- A script to output the efficient frontier which is a data frame containing 100 portfolios with their risks and returns.
- A script to output the weights of the assets making a certain portfolio after the portfolio manager has selected the one they were interested in from the previous script.
- A script for each one of the assets showing the Arithmetic mean, Standard deviation, Geometric mean, and beta value. (These values are all computed using the closing price of the asset)
- A script for each one of the assets showing a summary for the projection resulted from the fitment of the historical data to the ARMA model. These summaries contain all of the information stated above under the “Forecast” function.
- A script for each one of the forecasts to plot the historical returns as well as the predictions for 25 periods on the same graph with the upper and lower boundaries and the two levels of confidence.

Using this output file with R studio will enable anybody, regardless of their level of acquaintance with R, to display the important information in an organized manner. Thanks to R studio’s split screen view, data frames and summaries and graphs will each appear on separate parts of the screen so there will be no confusion.
9. Conclusion

This project equips everyday portfolio managers and the general public as a whole with a tool that brings to them tools and techniques they would normally have study a lot for. This program doesn’t bring-in any new technology, nor does it develop new techniques in the areas of portfolio management and mathematical modeling. The work that was performed is to learn about tools and techniques available from different areas and try to combine them into one program. A lot of research was done in the areas of Finance, math and statistical programing languages. Once the tools that fit best the needs of this project were selected, the need became to blend them all together in a simple and easy to understand program and use by everybody. As shown in the demo, by only following simple instructions and answering a couple of question, the user of this program gets to perform a simulation on stocks and stock returns as well as fit their data to a mathematical model and do projections for future returns. Without this program the person that wants to get similar results would have to have had some sort of training in a multitude of fields. In addition to that using a programing language like R cuts off the running time by a considerable amount. This is just the first step in what could be a project of huge wingspan. Future work includes providing the user with more choices on what constraints to put on their portfolios, as well as providing them with other mathematical models so that they can choose the one that fits their needs the best (provided that they have a strong mathematical background).
10. References


11. Appendix 1: Demo

The following section will follow the execution of the program step by step as if it was deployed and used on industry by a portfolio manager. It will be written just like a user-guide documenting every action that needs to be accomplished by the user. The following will also include screenshots to give the reader a visual aid and familiarize him/her with the environment.

First the user should launch the R-studio IDE, once that is done him/her will be presented with the following interface:

![Figure 4: The R Studio interface](image)

After that the user should load the “.R” file containing the R-scripts in order to run the program. That could be done by clicking on File >> Open File and then locating the file wherever it’s
stored on the machine. In the case of this demo the file is named “draft2.R” and is located under the file “Capstone”:

![Figure 5: Opening the R script on R studio](image)

Once that task is completed the file should be loaded in the top left corner of the screen and can be executed using the “run” button right above it.

After the program is executed; it will first proceed by loading all the libraries that are required for its functioning; that is done because the program relays heavily on packages and their functions.
Figure 6: Loading the libraries to the current session

Once all the libraries loaded the program asks the user for the first input. That is the number of assets that they want to invest in. The maximum number is now limited to 10 because of showcasing purposes, it can however be upgraded to however much the client wants. For our demo the number of assets will be set to 5.

Figure 7: Prompt for the number of assets to invest in
After entering the number of assets one wished to invest in, they are presented with a prompt for a starting and ending dates. The user should enter the start as well as end date from which they wish to start pulling historical prices for their assets. For the purposes of our demo let the start date be “2010-10-10” and end date be “2017-03-03”:

Figure 8: Prompt for the start and end date for data retrieval

After entering the dates, the user is asked to enter the names of all the assets as well as the index they all belong to, as they appear on the stock market. Meaning that the user should enter the abbreviation rather than the name of the company. For the purposes of this demo let us enter the stock prices for IBM, Google, Apple, Yahoo and Amazon. For the index let us enter, NASDAQ-100 technology sector index:

Figure 9: Entering 5 stocks and their index

Once the user has entered all of the stock names as well as the index to which they belong, the historical prices are fetched on an OHCL format. Meaning that for each stock the program pulls for each day (from the start date to the end date) the following:

-Open price
-Highest price
-Lowest price
- Closing price
- Volume of trading
- Adjusted price

For our example, the IBM stock prices will look like this:

<table>
<thead>
<tr>
<th>Date</th>
<th>IBM.Open</th>
<th>IBM.High</th>
<th>IBM.Low</th>
<th>IBM.Close</th>
<th>IBM.Volume</th>
<th>IBM.Adjusted</th>
</tr>
</thead>
<tbody>
<tr>
<td>2010-10-11</td>
<td>138.79</td>
<td>139.94</td>
<td>138.64</td>
<td>139.66</td>
<td>4004300</td>
<td>119.1873</td>
</tr>
<tr>
<td>2010-10-12</td>
<td>138.40</td>
<td>139.99</td>
<td>138.27</td>
<td>139.85</td>
<td>5637300</td>
<td>119.3494</td>
</tr>
<tr>
<td>2010-10-13</td>
<td>139.91</td>
<td>141.48</td>
<td>139.78</td>
<td>140.37</td>
<td>8784300</td>
<td>119.7932</td>
</tr>
<tr>
<td>2010-10-14</td>
<td>140.35</td>
<td>141.50</td>
<td>139.69</td>
<td>141.50</td>
<td>5653100</td>
<td>120.7576</td>
</tr>
<tr>
<td>2010-10-15</td>
<td>142.10</td>
<td>142.10</td>
<td>140.54</td>
<td>141.05</td>
<td>7220400</td>
<td>120.3821</td>
</tr>
<tr>
<td>2010-10-18</td>
<td>140.90</td>
<td>143.03</td>
<td>140.84</td>
<td>142.83</td>
<td>10322700</td>
<td>121.8926</td>
</tr>
<tr>
<td>2010-10-19</td>
<td>137.30</td>
<td>139.34</td>
<td>136.70</td>
<td>138.03</td>
<td>14159100</td>
<td>117.7962</td>
</tr>
<tr>
<td>2010-10-20</td>
<td>138.28</td>
<td>139.87</td>
<td>138.13</td>
<td>139.07</td>
<td>6945500</td>
<td>118.6838</td>
</tr>
<tr>
<td>2010-10-21</td>
<td>139.81</td>
<td>140.49</td>
<td>138.40</td>
<td>139.83</td>
<td>6221200</td>
<td>119.3324</td>
</tr>
<tr>
<td>2010-10-22</td>
<td>140.15</td>
<td>140.75</td>
<td>139.46</td>
<td>139.67</td>
<td>4105300</td>
<td>119.1958</td>
</tr>
<tr>
<td>2010-10-25</td>
<td>140.42</td>
<td>141.40</td>
<td>139.81</td>
<td>139.84</td>
<td>4870000</td>
<td>119.3409</td>
</tr>
<tr>
<td>2010-10-26</td>
<td>139.29</td>
<td>142.00</td>
<td>138.53</td>
<td>140.67</td>
<td>7327100</td>
<td>120.0492</td>
</tr>
<tr>
<td>2010-10-27</td>
<td>139.52</td>
<td>141.57</td>
<td>139.00</td>
<td>141.43</td>
<td>6465300</td>
<td>120.6978</td>
</tr>
</tbody>
</table>

Figure 10: IBM OHCL data frame

The next thing that is asked from the user is to say whether or not they want to enter a constraint on the portfolio that they are developing. It is important to note that some constraints such as box and full investment have already been added to the portfolio. If the user answers “yes”, he is presented with two choices, either enter a diversification constraint or overwrite the box constraint. For our demo let’s just use the constraints that are enforced by default by the program. This part of the program looks like this:

![Figure 11: Prompt for constraints to be added to the portfolio](image)
After that the input from the user is done, the program then does its work all while issuing some warnings if there are any. For our example the “suggestions” and “side-information” provided by the program tell us that some constraints are restrictive and make the program let drop some alternatives, and also that the mathematical model shows a possible convergence problem. These were accounted for and shouldn’t affect the results provided by the program:

Figure 12: Suggestions and warning messages from the program

Once all of this was done, the user should access the main results that were issued by the program. In order to do that the user should load the “output.R” script just like we loaded the “draft2.R” earlier.

This file contains a detailed description and a script for each one of the results that is useful to the portfolio manager.

The first one graphs all the feasible portfolios from those that were generated showing where the efficient frontier stands:

Figure 13: Script to visualize the portfolios plot with Risk VS Return
The resulting graph is the one that was made with “Plotly”, and using the code provided in their website. For our example the resulting plot looks like this:

![Random Portfolios with Plotly](image)

Figure 14: Random portfolios plot

The efficient frontier is also available for the user to see. It contains 100 portfolio risks and returns and is conveniently presented under the form of a table. In our demo that table looks like this:

```
5  # To visualize the efficient frontier's returns and risks select and run the line below
6  View(eff.frontier)
7
```

Figure 15: Script to visualize efficient frontier

The result after running that line is this:
Once the user has decided on what portfolio they want to invest on, they can run the script right after it in order to get the weights of the assets making up that portfolio. The script looks like this:

```
# To visualize the corresponding asset weights select and run the line below
View(frontier.weights)
```

Once that script executed, the result looks like this:
If the user knows what portfolio they are interested in and don’t want to see the weights of all 100 portfolios they can run this following script:

```r
# If you have selected a portfolio you want please run the lines below and enter
# the correspondent row number of the portfolio you chose
frontier.weights[as.numeric(readline("please enter the row number: ")),]
```

Figure 19: Script to retrieve weights of assets making up a specific portfolio

Once that script is selected and run, the user then is prompted for a row number form the efficient frontier data frame and once that number is entered the program displays the corresponding weights of each asset that make up that portfolio. For our example, let’s say that we decided to invest in the 5\textsuperscript{th} portfolio from the efficient frontier data frame:

```
> frontier.weights[as.numeric(readline("please enter the row number: ")),]
please enter the row number: 5
AAPL   GOOG   IBM   YHOO   AMZN
0.05000000 0.06440806 0.46462419 0.19534364 0.22562411
```

Figure 20: Results from single portfolio weights retrieval

A portfolio manager will not only use the results of the simulation and relay blindly on them. Some of the values that were manually computed and that can be used by the portfolio manager are:

- The arithmetic mean
- The standard deviation
- The geometric mean
- The beta value
The code for displaying these results looks like this:

```
11 # To visualize the Arithmetic mean, Standard Deviation, Geometric mean, and Beta value
12 # of all the assets you have selected select and run the following lines (one by one)
13 # PS: Run only up to the number of assets entered at the beginning
14 # Line 1 is for the first asset you entered and 2 for the second and so on ...
15 View(asset1values, as.character(asset[1]))
16 View(asset2values, as.character(asset[2]))
17 View(asset3values, as.character(asset[3]))
18 View(asset4values, as.character(asset[4]))
19 View(asset5values, as.character(asset[5]))
20 View(asset6values, as.character(asset[6]))
21 View(asset7values, as.character(asset[7]))
22 View(asset8values, as.character(asset[8]))
23 View(asset9values, as.character(asset[9]))
24 View(asset10values, as.character(asset[10]))
```

Figure 21: Script for retrieving manually computed values for individual stocks

For our demo the results look like the following for the IBM stock:

<table>
<thead>
<tr>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0002074253</td>
</tr>
<tr>
<td>2</td>
<td>0.0119743324</td>
</tr>
<tr>
<td>3</td>
<td>0.0001353764</td>
</tr>
<tr>
<td>4</td>
<td>0.6108439713</td>
</tr>
</tbody>
</table>

Figure 22: Values for the IBM stock

After that the output file provides the user with scripts to visualize the results from the forecasts made by the fitting of the Autoregressive moving average model. The scripts look like this:
Figure 23: Script for retrieving the automatically generated summary for each stock

If we run the first one it will give us the results for the IBM stock, those look like the following:

Figure 24: Summary for the IBM stock
This summary contains most of the relevant information that resulted from the use of the ARMA model including the range values that were predicted for every return and the level of confidence (80% and 95%) with high and low values for both.

Finally, and after all of that has been done, the output file provides the user with a plotting function in order to plot the historical returns along with the predictions made by the model. That portion of the code is the following:

```plaintext
# To visualize the plot of the ranges of predictions for the next 25 days for
# a certain asset select and run the following lines (one by one)
# PS: Run only up to the number of assets entered at the beginning
# Line 1 is for the first asset you entered and 2 for the second and so on ...
plot(arma.forecast1, main = "ARMA forecasts 1")
plot(arma.forecast2, main = "ARMA forecasts 2")
plot(arma.forecast3, main = "ARMA forecasts 3")
plot(arma.forecast4, main = "ARMA forecasts 4")
plot(arma.forecast5, main = "ARMA forecasts 5")
plot(arma.forecast6, main = "ARMA forecasts 6")
plot(arma.forecast7, main = "ARMA forecasts 7")
plot(arma.forecast8, main = "ARMA forecasts 8")
plot(arma.forecast9, main = "ARMA forecasts 9")
plot(arma.forecast10, main = "ARMA forecasts 10")
```

Figure 25: Script to plot the historical return and the predicted ones

Once the first script is executed in our demo, it will issue the following graph for the IBM stock. The black part is the one representing the historical returns and the blue parts represent the predictions with the two ranges of confidence.
ARMA forecasts for INFY returns

Figure 26: Plot of the historical returns and predicted ones
12. Appendix 2: The accuracy of the model

As is stands right now, this program is in “showcase” mode. The way the predictions are made is by fooling the computer into thinking that the provided data is all there is. That way we keep some actual results to test our predictions against. The whole purpose of this is to judge the performance of the ARMA model in predicting stock returns. There is no good or bad mathematical model, but some are better suited than others for specific tasks.

We can then check the accuracy of our model by calling the “accuracy” function and giving it as input the model’s predictions and the actual test set we had saved before.

```python
accuracy(arma.preds, infy_ret_test)
```

The accuracy function returns multiple measures, the one I have reference data from is the RMSE, so it will be the one I will use.

RMSE stands for Root Mean Square Error and it’s used to assess the differences between the values predicted by a model and the actual values. In this example, here is the output of the function:

```
> accuracy(arma.preds, infy_ret_test)

ME   RMSE    MAE   MPE   MAPE
Test set 0.07837922 1.061067 0.7524204 98.20713 100.5867
```

Figure 27: Accuracy measures of predicted VS actual returns

We can see that our RMSE is of 1.061067 which is a relatively small value. Based on that we can consider our predictions to be reliable enough to be trusted.