

**Portfolio Optimisation   
using Modern Portfolio Theory & Python**

Graeme Hutchison

HUT14470469

Project Supervisor:  
 Bowei Chen

**Acknowledgements**

**Abstract**

Contents

[1. Introduction 5](#_Toc480394731)

[1.1 Background 5](#_Toc480394732)

[1.2 Aims & Objectives 6](#_Toc480394733)

[Aim & Rationale 6](#_Toc480394734)

[Objectives 6](#_Toc480394735)

[2. Literature Review 7](#_Toc480394736)

[2.1 Modern Portfolio Theory 7](#_Toc480394737)

[Background 7](#_Toc480394738)

[Model Inputs 7](#_Toc480394739)

[Diversification 8](#_Toc480394740)

[Portfolio Performance Evaluation 9](#_Toc480394741)

[3. Methodology 10](#_Toc480394742)

[3.1 Project Management 11](#_Toc480394743)

[Time Planning 11](#_Toc480394744)

[Risk Management 11](#_Toc480394745)

[Development Tools 11](#_Toc480394746)

[4. Implementation 16](#_Toc480394747)

[4.1 First Iteration – Retrieving and Handling Data 16](#_Toc480394748)

[Stock Pricing Data 16](#_Toc480394749)

[4.2 Second Iteration - 16](#_Toc480394750)

[5. References 17](#_Toc480394751)

**Figures**

[Figure 1 - The effects of Diversification in different markets (Solnik, 1995) 8](#_Toc480394717)

[Figure 2 - Graph showing the popularity of languages found in Data Science job listings. (Muenchen, 2017) 13](#_Toc480394718)

[Figure 3 - Stackoverflow's developer survey - Most popular Version Control (Stackoverflow, 2015) 14](#_Toc480394719)

[Figure 4 - Iterative version of SDLC. (Powell-Morse, 2016) 16](#_Toc480394720)

# Introduction

## Background

This section aims to set the scene of the project, providing context and introducing the problem at hand. An investment is defined as “The Action or process of investing money for profit” (Oxford Dictionary, 2017). There are lots of different options that investors are faced with when it comes to choosing what to invest in. There are stocks, bonds, options, futures and countless more to consider. For this project, the focus is going to be on stocks. A stock is a portion of shares issued by a company (Oxford Dictionary, 2017). When investing into stocks, or anything for that matter, the general goal is going to be that when it comes time to sell, they have risen in value enough that the investor has made a profit. However, making that profit almost never comes free. Each investment typically has an “Expected Return” and a “Risk” aspect associated with it. These can be defined in a number of ways. In the context of this project, “Expected Return” is defined as the anticipated profit or loss on an investment characterised by a historical average. “Risk” is defined as the volatility of an investment’s return, which is effectively the likelihood that an investment’s actual return will be different to the expected return. This is characterised as variance. The best possible investment would be one that had a very high return with no risk attached. Sadly, this is never the case as almost every investment is guaranteed to have some kind of inherent risk (Thangavelu, 2015). To achieve those higher returns, investors have to be prepared to take on the extra risk involved in doing so.

It would be unusual for investors to only invest in one asset at a time Instead, what is created is a portfolio. A portfolio is “a range of investments held by a person or organisation” (Oxford Dictionary, 2017). Due to the insanely large amount of asset options that are available for investors to choose from, this introduces a small part of the problem that this project addresses. How do you decide what assets to invest in? Traditionally when creating portfolios, investors would select assets using anecdotal analysis or statistics of each asset individually. This changed when Harry Markowitz’s paper “Portfolio Selection” was released in March 1952 compiled within The Journal of Finance. Markowitz put forward among others, the idea of looking at how assets move and interact with the portfolio as a whole rather than on an individual basis. Instead of looking at an asset’s individual variance, it instead considers the risk of the portfolio to be determined by the covariance between each asset. In finance, covariance is a measure of how much the returns of two assets move together, contextualising the risk of a specific asset with the rest of the portfolio (Investopedia, 2017). One of Markowitz’s biggest ideas is that investors can potentially decrease the amount of risk they are having to take for the same return through means of diversification (Shipway, 2009). By spreading investments out over a larger number of diverse assets, the risk associated is generally going to be lower than if the investments were made in only one or two assets. This is due to each asset reacting differently to certain scenarios. With a large and diverse portfolio, there are going to be scenarios in which some assets will pay off, and some that won’t (Elton and Gruber, 1997).

The age old question of how to distribute your budget between each of your assets forms the problem that this project aims to provide a solution for. Modern Portfolio Theory (MPT) is a combination of Markowitz’s “Portfolio Selection” theory and William Sharpe’s later contributions in 1964 with his Capital Asset Pricing Model (CAPM). It is an investment framework that provides guidance for the creation of portfolios based on the maximisation of expected returns for a given risk (Mangram, 2013). Markowitz said that for every level of risk, there exists what he described as an “efficient portfolio”. This being the portfolio that provides the maximum return for that specific amount of risk. The Mean-Variance analysis techniques featured within Modern Portfolio Theory provides insight into mathematical ways of identifying and calculating efficient portfolios through distribution of budget, optimising the trade-off between risk and return. Despite its reputation, original forms of Markowitz’s optimisation model aren’t extensively used for larger portfolios due to the computational complexity involved in optimising them (Konno and Yamazaki, 1991). Along with the larger advancements of computing into the financial world, optimisations like these are becoming easier to do, due to strides in both software and hardware. It is this area in which the project will take place; using mean-variance analysis to craft and optimise select portfolios to achieve the desired goals be it; minimum variance, maximum return or somewhere in between.

## Aims & Objectives

### Aim & Rationale

The aim and rationale of this project is to create a proprietary asset allocation system that demonstrates the effectiveness of Modern Portfolio Theory in creating investment strategies. It should be able to retrieve relevant and up-to date stock data providing visualisation and descriptive analysis to the user. It will allow the user to craft a portfolio using their choice of stocks from the available selection. Using concepts from Modern Portfolio Theory such as Mean-Variance analysis, this portfolio can then be optimised in order to achieve the user’s desired goals. The statistics of the portfolio’s potential performance will be displayed by the system so the user then decide whether to proceed with that portfolio as their investment strategy.

### Objectives

The objectives of this project are to:

* Research and develop a greater understanding of Modern Portfolio Theory to help pinpoint requirements and guide implementation.
* Find and consider various options for a reliable online source of relevant financial data.
* Implement solutions using Python to achieve the following:
  + A retrieval system to fetch up-to-date price data for a specified list of stocks.
  + Perform statistical analysis on the data to provide the user with descriptive summaries of each stock as well as graphic visualisation.
  + Allow the user to craft custom portfolios manually choosing from the available selection.
  + Perform mean-variance optimisation to achieve the user’s desired portfolio goal.
  + Provide the user with summarising statistics of their optimised portfolio.
* Create a full graphical user interface to contain and situate the functionality of the solutions that have been created to form an interactive desktop application.

# Literature Review

This chapter illustrates and explains some of the underlying concepts of this project through the review of relevant academic work. Description and analysis of their work is performed, furthered by presenting a comparison of their findings with work of a similar nature.

## Modern Portfolio Theory

### Background

The original founder of Modern Portfolio Theory (MPT): Harry Markowitz had his paper “Portfolio Selection” first published in the Journal of Finance, March 1952. His work and theories provided completely new and revolutionary insight into many areas of finance, totally changing the traditional methods of managing investments, awarding him a Nobel Prize in Economic Sciences in 1990. (Elton and Gruber, 1997). He characterised and formulated the problem of financial portfolio selection as a more than just a consideration of return, but as a combination of both; risk and return, suggesting that there is an unavoidable trade-off to be made. MPT assumes there to be a positive relationship between risk and return for each investment, meaning that to obtain a higher expected return, there has to be a higher risk involved. This introduced the concept of mean-variance analysis into portfolio theory, called the Markowitz Model. The Markowitz model can also be called the mean-variance model because it takes the expected return as a mean and uses variance of standard deviation as a measure of risk. It explains that for every level of given risk in a portfolio, there is what Markowitz describes as an “efficient portfolio”, aka one portfolio that provides the greatest return for that level of risk. This laid the foundation for other theories such as the “Capital Asset Price Model” which extended the analysis of determining what assets should be included into a portfolio (Elton and Gruber, 1997).

### Model Inputs

A common way to calculate inputs for the Markowitz model is to look at historical data as a means to estimate values for sample mean for expected return and variance for risk. There is contention about the accuracy and effectiveness of using historical data as inputs for the mean-variance optimisation model. Fabozzi et al (2002) discuss the effectiveness of using historical data as an input, stressing that there is a strong level of importance in assessing the economic conditions when using that period to estimate return and variance. It goes on to suggest that there are reasons for the use of historical data, including arguments about the length of the histories, as well as the political and financial state of that market playing a major role in assessing the viability of the historical data. They argue that only after an economy has a long record of good performance and stability before should they be considered for use as an estimation. To contrast this point, Michaud (1989) created a similar implementation of a mean-variance optimiser using historical data. He theorised that the poor performance of the model was due to the use of historical data. Going on to suggest that a trait of the optimiser was causing it to maximise the error contained within the data due to what he believed was because the estimations made ignored what he described as the “inherent multivariate nature” of the problem.

Furthering the discussion of effective inputs for the Markowitz model, Konno and Yamazaki (1991) question the use of standard deviation of historical returns as a measure of risk. They note that an investor’s perception of risk is not symmetrical around the mean of the returns. Arguing that investors aren’t as concerned when there is large positive deviation from the mean compared to small negative deviations and going further to say that market returns are generally not distributed symmetrically. Rockafellar and Uryasev (2000) assess the use of alternatives to standard deviation in optimisation practices, recommending the use of more intricate methods to characterise risk, such as Value-at-Risk (VaR), or even Conditional Value-at-Risk (CVaR). Both of which are based on the use of standard deviation of normal distributions and contain undesirable attributes described as a “lack of subadditivity and convexity” making optimisations more computationally difficult as well as an increase in implementation technicality.

### Diversification

The concept of diversification is one of the most important parts of Markowitz’s work. Even today, the concept of diversification is commonly used in portfolio management, as it such an intuitive appeal as well as an empirical backing. Diversification is the process of spreading investments over a wide range of investment types and areas in an attempt to reduce unsystematic risk (Markowitz, 1952). There are two types of risk that the financial world. There is systematic and unsystematic risk, both of which are a component of the overall risk that is present within assets and portfolios. Systematic risk can be defined as the risk that is inherent to an entire market, meaning that it is something that all assets that are traded within that market are subject to. An example of Systematic Risk would be the effects applied on the whole market by events such as The Great Recession, not the event itself. Events that have these affects are what is known as Systemic Risks (Investopedia, 2017). Unsystematic Risk is the part of an investment’s risk that is specific to that investment, influenced by events attributable to the investment itself or the sub-group it is in. Not the entire market system like Systematic Risk (Investopedia, 2017).

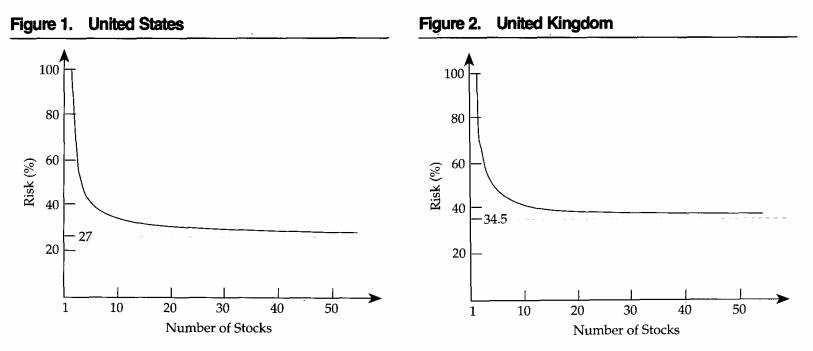
The effectiveness of Diversification at reducing unsystematic risk is generally undisputed. Solnik (1995) analysed the effects of diversification on markets outside of the US finding that it was almost just as effective in European markets. However, he noted that the proportionality of the number of assets compared to the amount risk reduced was not consistent and it was impossible to completely eliminate risk after a certain point because of what can be attributed to the assets moving together. He suggested that it is possible to reduce risk past this certain point through further diversification extending the holding of assets across multiple markets rather than just one. 

Figure 1 - The effects of Diversification in different markets (Solnik, 1995)

Raffestin (2014) builds on the summary provided by Solnik (1995), investigating the effectiveness of Diversification at reducing the impact caused by systemic risks, theorising that Diversification can make investors safer individually but can actually increase the effects caused by a systemic risk due to common asset holdings between each investor. He described the connections made by these common asset holdings as “endogenous covariance” that can actually propagate the effects of systemic risk through the system. He found again, that investors can minimise their risk by spreading wealth across not only a greater amount of assets, but more distant assets, potentially increases the overall stability of a market. This further reinforces the importance of strong Diversification in a portfolio.

### Portfolio Performance Evaluation

There are a number of different metrics to measure the overall performance of a portfolio.

* Portfolio Evaluation methods
  + Use of market as benchmark
  + Index funds

# Methodology

Every project will usually have some kind of project management and software development methodology that it will follow. For very small projects, methodologies aren’t really much of a concern, as they generally don’t last long enough or aren’t intense enough to utilise the full benefit of following an established software development methodology. However, as the project grows in size, it becomes more and more important to be following a methodology as it can make time and cost more foreseeable, as well as boosting efficiency by sticking to a thought-out schedule (Awad, 2005). This makes it important to consider a range of methodologies for both project management and software development. Before choosing the most optimal methodology, relevant analysis would need to done for this particular project.

All of the software development methodologies out there each have their own strengths and weaknesses that can become more or less apparent depending on the type of project. One of the most influential concepts that affects the viability of a software development methodology is the initial state of the software’s requirements (Balaji and Murugaiyan, 2012). The way the requirements were set out at the beginning was very uncertain. The artefact being produced from the project was always going to be making use of various concepts within Modern Portfolio Theory (MPT). As MPT is a relatively old theory, all concepts featured within it are already laid out and are not going to change. This makes it seem possible to create a full list of requirements from what had been researched, and proceeding with a traditional sequentially structured methodology such as Waterfall. However, due to the sheer amount of concepts situated within MPT, it is extremely hard at the beginning of the project to fully gauge how many of these concepts were realistically going to be implemented, and in general, how far the artefact could go. The project supervisor acted not far from what could be considered a type of client to the project. The direction and requirements that were dictated by it were often the result of in-depth discussion with the project supervisor. This set a slight feature-driven-development tone to the project as the outcomes of these discussions provided both general direction and priority. This uncertainty alone would rule out the use of a sequential methodology like Waterfall. It tends to benefit from having very well defined requirements, as well as a strong time and cost forecast which would be unable to accommodate this type of project (Balaji and Murugaiyan, 2012).

Going off the exclusion of rigid methodologies such as Waterfall, the logical choice is something a lot more flexible that could adapt to the initial lack of knowledge in the chosen field, making determination of detailed requirements and calculating an accurate time plan for the project impractical. This lent itself to choosing a more adaptable methodology that could account for these unknown details, providing a framework for both software development and project management. In terms of project management, some of the more intricate and established methodologies such PRINCE2 seemed a little too process heavy and intensive for a solo project. Something such as PRINCE2 is more suitable and effective for a larger team, where allocating people and resources is more of a complex problem. It would be unnecessary and perhaps detrimental to try incorporate this large of a methodology for a small project like this (Matos and Lopes, 2013).

An Agile methodology would be most fitting for a project of this specification. It provides multiple opportunities to assess the overall direction the project is heading in, which accommodates the uncertainty of this project’s final goal. The iterative and incremental nature of Agile makes it extremely appealing as there are regular intervals in which new requirements can be gathered and work done previously can be built on and improved (Agile Methodology, 2008). While Agile itself is not a methodology, there are methodologies that use Agile principles. One of the most popular ones is SCRUM (SCRUM, 2017). One of the problems with using SCRUM, is that it is so heavily focused on team communication and feedback, that a lot of its benefits are wasted on a one-man project. For this project, the chosen methodology is Agile in nature, primarily combining aspects of Extreme Programming (XP) and Feature-Driven Development (FDD). XP accommodates the heavy software development aspect of the project as well as the presence of changing requirements and FDD relates to the prioritising of features to include in the final release of the artefact.

## Project Management

### Time Planning

Time planning is something that is essential to any project’s success. However, since the project follows an Agile methodology it is difficult and often useless to try and plan every part of the project accordingly. The project is prone to changing requirements and deadlines throughout, making the process of accurately planning out each stage on a weekly basis a troublesome task. A Gantt chart is a commonly used time planning tool that provides good visualisation of how time is going to be spent over the course of the project (Investopedia, 2017). As the project follows an agile methodology, a hypothetical scenario of how the project is likely going to be structured has been generated. With the use of iterations, the generated Gantt chart details rough estimations of each iteration’s length and contents.

<Gantt Chart Here>

### Risk Management

As the project contained multiple software based components, it was important to consider some of the different risks that were involved. An essence of project management that could be utilised to summarise and handle this risk is what is called a Risk Matrix. A Risk Matrix can be defined as a structured approach to identifying risk, defining which risks are most dangerous to the project, what impact they might have, and how they can be dealt with (Garvey and Lansdowne, 1998).

A Risk Matrix has been formulated to provide evaluation and insight into potential risks that may be present within the project. <Talk about what is included in Risk Matrix>

<Risk Matrix Here>

### Development Tools

This section delves into the main tools and applications that were used in the process of creating the project’s final artefact. It explains what specific tools were used for, how they were chosen and if applicable, what were some possible alternatives.

#### Programming Language

One of the most important tools utilised in a software development project is the programming language that it is created in. There are programming languages that are much more suited to specific tasks than others and it’s helpful to pick a suitable one so as to not make things purposely difficult or inefficient. When deciding what programming language is going to be used, it is important to classify what kind of problems are needing to be solved using language. A brief overview of the tasks in this project involve a series of data analysis and handling techniques including fairly comprehensive mathematical procedures. Finally, the creation of a graphical user interface (GUI) is also necessary, which really requires some kind of object-oriented principles.

Based on the data handling and mathematical function requirements alone, some potential candidate languages would be Python (Python Software Foundation, 2017) and R (R Core Team). R is one of the most popular languages used in data science and provides a large majority of the utilities needed for the project. The R environment contains an integrated suite of facilities for data manipulation, calculation and graphical display (R Core Team, 2017). However, R unfortunately does not contain the object oriented features that would be necessary to create a GUI. Python on the other hand, was not initially designed as a data analysis tool. However, due to its general applicability, several high quality modules have been developed to handle and manipulate data, providing similar functionality to R. This has made Python extremely popular in the Data Science world and a real competitor to R (Muenchen, 2017). This makes Python the ideal choice for this project due to its versatility as it can perform everything necessary to assure its completion. There are quite a few different distributions of Python, each coming with their own pre-packaged modules. Since this project is fairly heavy on mathematical and data handling processes, Anaconda (Continuum Analytics, 2017) was the chosen distribution as it already contained a lot of the necessary Python modules with it, making it the logical choice.



Figure 2 - Graph showing the popularity of languages found in Data Science job listings. (Muenchen, 2017)

#### Integrated Development Environments (IDE)

IDEs can be fairly important when it comes to developing software. They package most of the fundamental tools that a developer is going to need into a single graphical user interface. This streamlines the process of writing code as it provides many quality of life improvements that can often be overlooked in terms of importance. Spyder (Spyder Developer Community, 2017) is a cross-platform IDE that is used for a lot of Python projects involving data processing and mathematics. Some of its main features that were utilised during the development process were things such as its integrated console. It has an interactive shell called IPython that provides support for data visualisation and also contains a flexible interpreter providing a quick and built-in way to run the program in development, making debugging and general program execution a lot faster. It provided instant, on-screen console output pertaining to the program as well as detailing bugs were they to occur.

Code completion also provides a huge boost to productivity, especially within object oriented projects. It helps by providing suggestions of relevant attributes pertaining to that specific class or object, as well as displaying information about that function’s parameters. This can become extremely useful when the project contains multiple instances of a complex class by making it easier to see how that class is set up.

A minor but important feature none the less, is syntax highlighting. Code can get difficult to read and understand when the size of the project begins to ramp up. Spyder contains settings that allow full customisation of syntax colours, providing easy ways to differentiate between various elements within code, such as functions, classes and data types. This greatly increases the overall clarity of the code by making its structure much more visually defined, making it easier to find errors stemming from specific areas of the code.

#### Version Control

With any project it is necessary to have some kind of centralised storage for all work created that it accessible from anywhere. Some storage systems are a lot more advanced and contain more effective forms of Version Control. A Version Control System (VCS) is a system that records all changes made to a set of files that are located within that storage system. It provides date and time specifications, as well as exact character changes that provides the user with the ability keep track of changes and recall older versions if necessary (Chacon and Straub, 2014). The benefits of using a VCS become more noticeable and apparent when doing a larger and more complex project. It becomes even more useful in a project such as this because of its iterative nature and use of Agile methodologies.

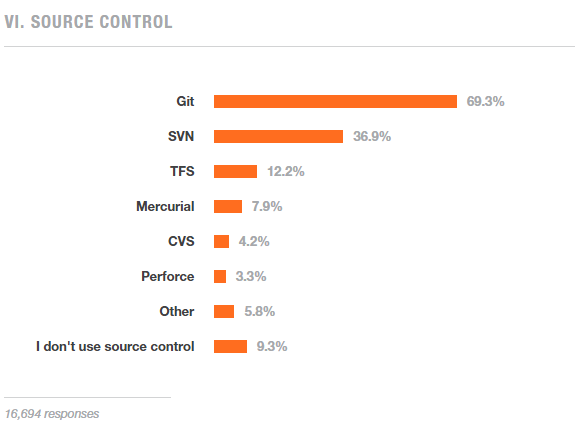


Figure 3 - Stackoverflow's developer survey - Most popular Version Control (Stackoverflow, 2015)

In 2015, Stackoverflow ran a survey for developers asking them a series of questions. One of the questions was their chosen VCS. Figure 2 shows the results from that survey which had almost seventeen thousand responses. The results show that roughly seventy percent of those developers used some version of Git (Hamano, 2017). This provided a solid indicator of which VCS is most popular and likely the most effective.

Git and an extension of that being GitHub (GitHub, 2017) was used for this project. University students get free access to a series of private online GitHub storage repositories that are ideal for a project like this as they have practically unlimited storage space. GitHub was used extensively throughout the project from the very beginning. Project work was often not done in the same place and usually on different systems. The work stored in the repository ranged from code used in development to diagrams and word documents as it was logical to keep everything in one place. The centralised nature of a system like GitHub provided ways to pull up-to-date versions of the project to any system and then push any changes that were made during that session back to the online repository. This made working on the project very flexible as it was extremely easy to access the repository from almost any location should it have been necessary. In conjunction with the push & pull system that GitHub incorporates, the amount of information that is available pertaining to each change made is extensive. This can help keep track of progress made during the project, as well as retaining old solutions that may have otherwise been discarded should they prove useful again.

#### Qt Designer

Creating the visual design and layout for Graphical User Interfaces (GUI) through pure coding practices can become an extremely time consuming process unless the exact look is solidified from the outset and not many changes are made. For projects where the visuals of the GUI aren’t the main aspect of the development, many developers will opt to use other software to accelerate and streamline the creation process. Qt (The Qt Company, 2017) itself is an application framework that runs on several different software and hardware platforms without requiring much code adaptation. Qt Designer is one of those applications that use this framework. It provides a toolkit containing useful features such as prebuilt widgets and layout structures to help users facilitate their application functionality in a timely manner (Blanchette and Summerfield, 2006). When it came designing and creating the layout of the GUI for this project, Qt Designer was used to handle almost all of the design tasks. It was extremely useful in the creation of the GUI wrapper as it saved a lot of time by automatically generating code for the layout designed within Qt Designer. Meaning that all the visuals of the application were instantly setup and all that needed to be done were tasks such as connecting inputs to specific functions and handling the outputs.

# Implementation

This section details the full development process for the artefact created during this project. It does not follow the traditional Software Development Life Cycle (SDLC) due to the Agile iterative methodology that has been adapted to this project. The development in this project is done over multiple iterations that repeat the typical stages found in the SDLC. The figure below roughly outlines the overall process used in the development of this artefact. Requirements were gathered before each iteration to figure out what was needing to be done. Suitable design was then generated based on those requirements which would then be implemented. Relevant testing to a varying degree would be performed at the end of each iteration to ensure that the result of that iteration was working as intended before starting the next. Some of the later iterations rely on and utilise functionality produced in the earlier iterations which makes it imperative that the dependencies found within the earlier iterations are fully complete. The project supervisor provided significant input when it came to gathering requirements and assessing the output from each iteration to ensure that it was time to begin the next iteration.

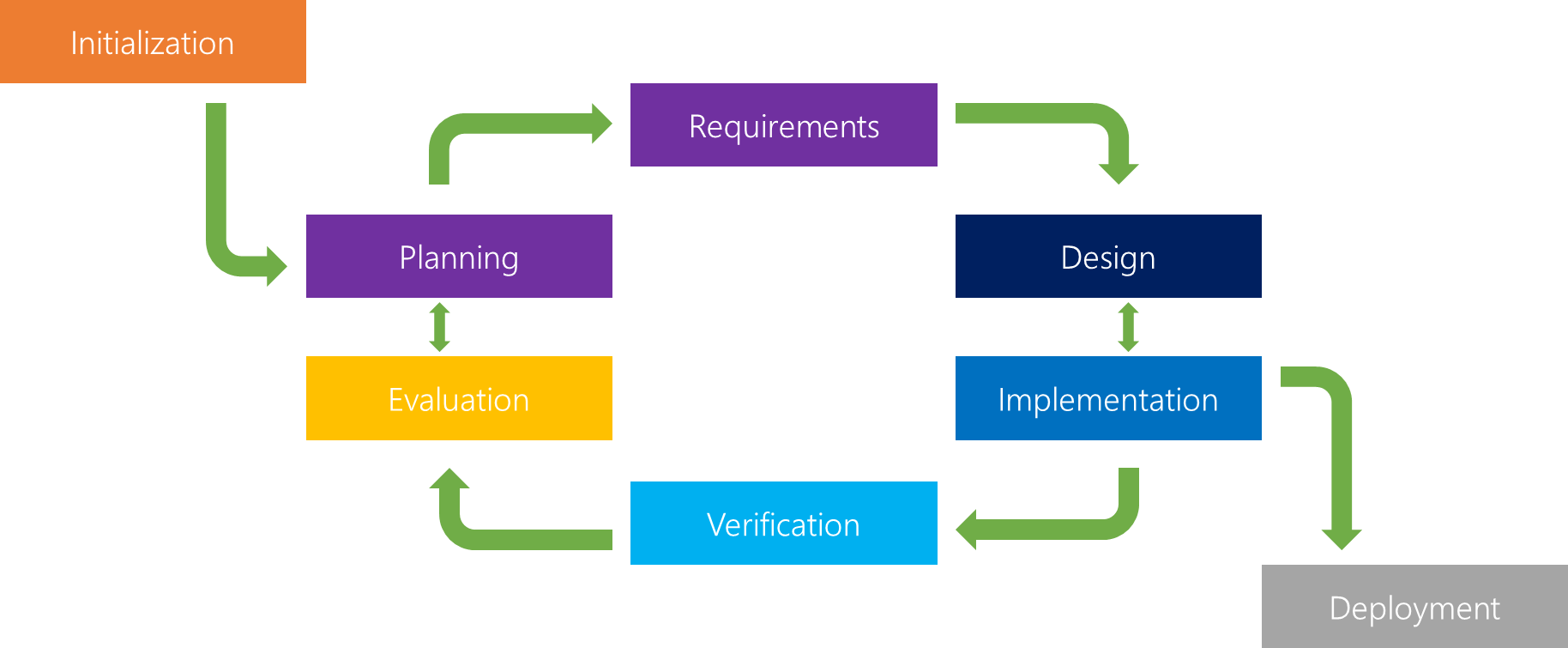


Figure 4 - Iterative version of SDLC. (Powell-Morse, 2016)

## First Iteration – Retrieving and Handling Data

### Stock Pricing Data

Modern Portfolio Theory and more specifically There are different metrics that can be used to calculate the expected return and risk of an asset. In this specific solution, the expected return and risk of an asset are going to be determined using statistical calculations on historical stock pricing data.

## Second Iteration -

# References