Mohammed VI Polytechnic University
AFRICA BUSSINESS SCHOOL

REPORT OF PROJECT

PORTFOLIO OPTIMIZATION USING LINEAR ALGEBRA

Major: Master in Quantitative and Financial Engeneering

SUPERVISOR(s): Prof. LAHCEN LAAYOUNI
—o0o—
MOUSSA AIT HAMZA

RABAT, APRIL 2023

Declaration

We guarantee that this research is our own, conducted under the supervision and guidance of Assoc. Prof. . LAHCEN LAAYOUN. The result of our research is legitimate and has not been published in any forms prior to this. All materials used within this researched are collected by ourselves, by various sources and are appropriately listed in the references section. In addition, within this research, we also used the results of several other authors and organizations.

Contents

List of Figures

Chapter 1

Introduction

Portfolio optimization is a fundamental problem in the field of finance, which involves identifying the optimal allocation of assets within a portfolio that maximizes returns while minimizing risks. In recent years, data science techniques and tools have increasingly been applied to this problem, enabling the development of more sophisticated and effective portfolio optimization strategies. By leveraging data analytics, machine learning, and other data science techniques, portfolio managers can gain deeper insights into the underlying assets, their associated risks and returns, and the relationships between them, and develop more robust and reliable optimization models. This has led to an increasing demand for professionals with expertise in both finance and data science, who can effectively apply these techniques to solve complex financial challenges. In this context, understanding the principles of portfolio optimization and how they can be applied in the context of data science is essential for anyone interested in pursuing a career in finance or data science.

1.1 Background and motivation

As a first-year master's student in quantitative and financial modeling, I have had the opportunity to study a variety of mathematical and financial tools that can be used to address the critical problem of portfolio optimization.

Portfolio optimization is a fundamental concept in finance that involves selecting the best mix of investments to maximize returns while minimizing risks. The optimization process requires a deep understanding of the underlying assets, their associated risks and returns, and the relationships between them.

By applying mathematical models and statistical methods, portfolio managers can make informed decisions about how to allocate assets and manage risk. With the knowledge gained from my coursework, I am excited to embark on a project with my fellow students to develop and implement a portfolio optimization strategy.

This project presents an opportunity to apply the concepts I have learned to a real-world problem in finance. The successful completion of this project will not only deepen my understanding of portfolio optimization but also equip me with valuable skills that are highly sought after in the finance industry.

Ultimately, the motivation behind portfolio optimization is to help investors achieve their investment goals while minimizing risks and maximizing returns, and I am eager to contribute to this important objective.

1.2 Objectives of the project

Portfolio optimization is a critical problem in the field of finance that has received a significant amount of attention in recent years. The goal of portfolio optimization is to identify the optimal allocation of assets within a portfolio that maximizes returns while minimizing risks. Achieving this goal requires a deep understanding of the underlying assets, their associated risks and returns, and the relationships between them.

To tackle this challenge, a group of first-year master's students in quantitative and financial modeling have embarked on a project to develop and implement a portfolio optimization strategy. The project objectives include applying principles of linear algebra and finance to the development of the strategy, using data science tools and techniques to analyze financial data and identify appropriate asset classes for inclusion in a diversified portfolio, and developing and implementing an optimization algorithm that leverages mathematical models and statistical techniques to consider key factors such as risk, return, and correlation.

The project also involves evaluating the performance of the optimization algorithm using both historical and simulated data, developing effective strategies for rebalancing and adjusting the portfolio to maintain optimal performance over time, and communicating the findings and recommendations of the project in a clear and concise manner, using data visualization tools and other techniques to make complex information accessible to stakeholders.

In addition to achieving these specific objectives, the project aims to leverage the principles of algebra, finance, and data science to develop a portfolio optimization strategy that is based on rigorous analysis and informed decision-making. It provides an opportunity for the students to develop skills in data analysis, modeling, and visualization, which are highly sought after in the finance industry, as well as to demonstrate the ability to work collaboratively in a team environment and apply problem-solving skills to complex financial challenges.

1.3 Overview of the methods and tools used

In portfolio optimization, QR decomposition and SVD decomposition are two important linear algebra techniques used to reduce the computational complexity of the optimization problem and identify the most important features of the dataset.

QR decomposition factorizes a matrix into an orthogonal matrix and an upper triangular matrix. This technique can be used to solve systems of linear equations and compute the inverse of a matrix. In portfolio optimization, QR decomposition can be used to reduce the computational complexity of the optimization problem by transforming the matrix of asset returns into an orthogonal basis.

Singular Value Decomposition (SVD) is another linear algebra technique used to factorize a matrix into its singular values and eigenvectors. SVD can be used to identify the most important features of the dataset and reduce the dimensionality of the problem. In portfolio optimization, SVD can be used to identify the most important factors that contribute to the returns of the assets in the portfolio.

Overall, QR decomposition and SVD decomposition are two important techniques used in portfolio optimization to reduce the computational complexity of the optimization problem and identify the most important features of the dataset. By applying these techniques, investors can optimize their portfolios more efficiently and achieve better returns while managing risk effectively.

Chapter 2

Literature Review

The literature review chapter is an essential part of this project as it provides a foundation for our understanding of portfolio optimization and its applications in finance. In this chapter, we will explore the existing literature on portfolio optimization and its relevance to the Moroccan financial market. We will also review the relevant literature on QR decomposition and SVD in portfolio optimization. The objective of this chapter is to provide a comprehensive overview of the theoretical underpinnings of portfolio optimization and its practical applications, particularly in the context of our project. By the end of this chapter, we aim to have a clear understanding of the current state of research in portfolio optimization and its applications in finance, which will serve as the basis for our methodology and analysis in subsequent chapters.

2.1 Portfolio Optimization

A portfolio investment involves holding a diversified range of financial assets, including stocks, bonds, real estate, and commodities, with the aim of increasing in value or generating returns over time. Investors can manage their portfolios independently or seek the assistance of financial advisors or money managers. The nature and composition of a portfolio investment are influenced by various factors such as the amount of capital to be invested, the investor's risk tolerance, and the planned investment period. There are two main types of portfolio investments:

- strategic investments that involve acquiring a set of financial assets to hold for the long term based on their potential for growth or yield income. This passive investment strategy involves maintaining assets over time and allocating them to investment classes using predetermined percentages established when the portfolio was first created.
- Active approach that refers to the investment strategy in which the main goal of the investor is to secure short-term gains by actively buying and selling the assets in their port-

folio. This strategy involves actively seeking out market opportunities to gain additional value. It is a more dynamic and proactive approach to investing compared to the strategic investment approach, which involves a more passive and long-term approach to maintaining assets and assigning them to investment classes based on predetermined percentages.

Portfolio Theory

The Modern Portfolio Theory (MPT), also known as the mean-variance analysis, is a theory that explains how risk-averse investors construct portfolios to maximize their expected return while taking into account the level of risk involved. This theory was developed by Harry Markowitz in 1952 and was later awarded the Nobel Prize.

Markowitz's theory argues that portfolios should be evaluated based on how risk and return affect the entire set of assets. The theory relies on statistical measures, such as variance and correlation, to assess portfolio performance. It suggests that investors should invest in multiple asset classes to achieve higher returns at a lower risk level. Diversification is the key to reducing portfolio risk compared to investing in a single asset. The mean-variance optimizer theory utilizes mathematical and statistical measures such as expected returns, standard deviations, and variance-covariance matrix to allocate the most efficient array of assets that will constitute the portfolio.

Portfolio Features and Metrics

Portfolio Return

The return on a portfolio represents the gain/loss resulting from the investment in various asset classes. Investors rely primarily on the historical returns to draw expectations about the future performance, and hence predict the expected returns. The portfolio's overall return can be computed using:

$$r_p = w_t \times r$$

Such that r_p represents the return generated by the portfolio, r is the column vector containing returns of each asset ri, and wt is the transpose of w, which is the vector containing the portfolio weights.

Portfolio Variance

The variance of a portfolio measures the dispersion of its returns and is calculated as the weighted average of the variances of the individual securities in the portfolio, plus the sum of all possible pairwise covariances between the securities, as follows:

$$\sigma_p^2 = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \sigma_{i,j}\rho_{ij} = \sum_{i=1}^{n} w_i^2\sigma_i^2 + \sum_{i=1}^{n}\sum_{j\neq i}^{n} w_i w_j \sigma_{i,j}\rho_{ij} \tag{2.1}$$

where σ_i is the standard deviation of returns on asset i, ρ_{ij} is the correlation coefficient between the returns on assets i and j, and w_i is the proportion of the total portfolio invested in asset i.

The Covariance Matrix

The covariance matrix is a mathematical tool used to measure the degree of association between the returns of different assets in a portfolio. To compute the covariance matrix, we use:

$$C = \begin{bmatrix} cov_{r_1,r_1} & cov_{r_1,r_2} & \cdots & cov_{r_1,r_N} \\ cov_{r_1,r_2} & cov_{r_2,r_2} & \cdots & cov_{r_2,r_N} \\ \vdots & \vdots & \ddots & \vdots \\ cov_{r_N,r_1} & cov_{r_N,r_2} & cov_{r_N,r_2} & cov_{r_N,r_N} \end{bmatrix}$$

$$cov_{r_i,r_j} = cov_{r_j,r_i} = E\left[(R_i - r_i) * \left(R_j - r_j\right)\right] w = [w_1, w_2, \ldots, w_N]^T$$

Such that $cov(r_i, r_j)$ represents the covariance between the returns of assets i and j. To compute the covariance, we use:

$$cov_(r_i, r_j) = E\left[(R_i - r_i)(R_j - r_j)\right]$$

The variance of individual assets is represented by the diagonal elements of the covariance matrix. A positive covariance indicates that the assets in the portfolio tend to move in the same direction, while a negative covariance indicates that the assets move in opposite directions. Generally, a lower correlation between assets results in lower volatility of the entire portfolio. This is why professional fund managers diversify their portfolios by selecting uncorrelated stocks, as the relationship between assets can significantly affect the overall risk of the portfolio.

The Efficient Frontier

Multiple portfolios can be constructed by an investor using a single assortment of assets by adjusting the weights assigned to each stock. The collection of all possible portfolios that have different weight arrangements is known as the efficient frontier, which was first introduced by Harry Markowitz. This method helps in identifying the group of investment portfolios that generate the highest returns for a given risk level. The efficient frontier plot, as shown in Figure 4, highlights the portfolios that have the maximum expected return for a particular risk level. Investors can use this plot to obtain various weight combinations based on their risk aversion and requirements. The black line in the plot represents the portfolio's efficient frontier, and portfolios located to the right of the plot exhibit a high level of risk. Conversely, portfolios located below the efficient frontier tend to generate lower returns. It is possible to identify the portfolio with the least risk and the one with the highest Sharpe ratio using this plot.

Therefore, the use of this plot helps in retrieving the optimal and efficient portfolio that complies with the degree of risk tolerance of each investor.

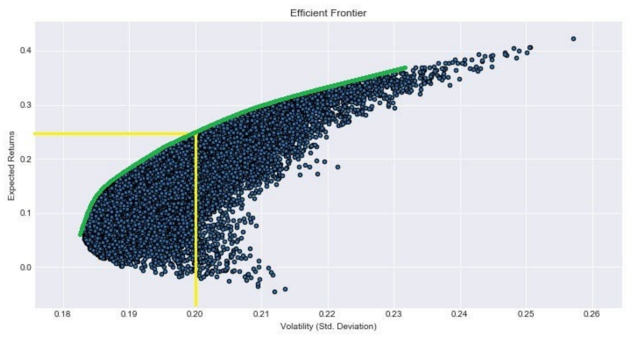


Figure 2.1: Portfolio optimization using the efficient frontier

Hence, the model that encapsulates the allocation and optimization of assets for an investor who is risk-averse can be expressed in the following manner: In this context, the matrix C stands

$$\boxed{\begin{array}{ll} \underset{\omega}{\text{minimize}} & \omega^t C \omega \\ \text{subject to:} & r_p = \omega^t r, \\ & \sum_{i=1}^{n} \omega_i = 1, \\ & \text{No short selling constraint, thus } \omega_i \geq 0, \\ & a < \omega_i < b \end{array}}$$

for the covariance matrix that describes the relationship between the returns of the various assets, while a and b denote the minimum and maximum weights assigned to each asset.

2.2 The covariance matrix and efficient frontier using SVD and QR decomposition

The QR decomposition and SVD decomposition can be used in a variety of ways after defining the covariance matrix and efficient frontier:

Computing the covariance matrix: The QR decomposition can be used to compute the covariance matrix of a set of random variables. Let X be a matrix of observations where each column represents a random variable. Then, the QR decomposition of X can be written as $X = QR$, where Q is an orthogonal matrix and R is an upper triangular matrix. The covariance matrix of X can be computed as $Cov(X) = (1/n)R'R$, where n is the number of observations.

Finding the efficient frontier: The efficient frontier is a set of portfolios that offer the highest possible return for a given level of risk. The SVD decomposition can be used to find the efficient frontier. Let X be a matrix of asset returns where each column represents an asset and each row represents a return for a particular time period. Then, the SVD decomposition of X can be written as $X = U\sum V^T$, where U and V are orthogonal matrices and $\sum$ is a diagonal matrix. The efficient frontier can be found by calculating the optimal portfolio weights for each level of risk using the singular values in $\sum$.

Portfolio optimization: The QR decomposition and SVD decomposition can be used to perform portfolio optimization. Given a set of asset returns and a covariance matrix, the QR decomposition can be used to compute the Cholesky decomposition of the covariance matrix, which can then be used to generate random samples of asset returns for simulation-based optimization. The SVD decomposition can be used to perform factor analysis and identify underlying factors that explain the variance in asset returns. These factors can then be used to construct portfolios that are more diversified and less sensitive to specific risk factors.

Chapter 3

Methodology

In this study, we will focus on the five stocks highlighted in the article mentioned here namely : Attijariwafa Bank, Maroc Telecom, Banque Centrale Populaire (BCP), Bank of Africa and LafargeHolcim Maroc ,and use data from the Casablanca Stock Exchange website to gather information on these stocks. Our objective is to construct an optimized portfolio consisting of these five stocks. To achieve this, we will employ two widely used mathematical techniques, namely QR decomposition and SVD. By applying these techniques, we aim to reduce the dimensionality of the data and extract the most important information to form a diversified and balanced portfolio that maximizes returns while minimizing risks. Our approach involves selecting stocks that complement each other and have a low correlation between them to reduce overall risk. We believe that the results of this study will provide valuable insights for investors who are looking to optimize their portfolio and achieve their investment objectives.

3.1 Finding the Global Minimum Variance Portfolio for Three assets

The global minimum variance portfolio $\mathbf{m} = (m_A, m_B, m_C)'$ for the three asset case solves the constrained minimization problem

$$\min_{m_A, m_B, m_C} \sigma_{p,m}^2 = m_A^2\sigma_A^2 + m_B^2\sigma_B^2 + m_C^2\sigma_C^2 + 2m_Am_B\sigma_{AB} + 2m_Am_C\sigma_{AC} + 2m_Bm_C\sigma_{BC} \tag{3.1}$$

$$\textbf{s.t. } m_A + m_B + m_C = 1 \quad \textbf{and} \quad \lambda = 1 - (m_A + m_B + m_C)$$

The Lagrangian for this problem is

$$\mathrm{L}(m_A, m_B, m_C, \lambda) = m_A^2\sigma_A^2 + m_B^2\sigma_B^2 + m_C^2\sigma_C^2$$

$$+2\, m_A m_B \sigma_{AB} + 2 m_A m_C \sigma_{AC} + 2 m_B m_C \sigma_{BC}$$
$$+\lambda\,(m_A + m_B + m_C - 1),$$

and the first order conditions (FOCs) for a minimum are :
$0 = \frac{\partial L}{\partial m_A} = 2m_A\sigma_A^2 + 2m_B\sigma_{AB} + 2m_C\sigma_{AC} + \lambda$, $0 = \frac{\partial L}{\partial m_B} = 2m_B\sigma_B^2 + 2m_A\sigma_{AB} + 2m_C\sigma_{BC} + \lambda$, $0 = \frac{\partial L}{\partial m_C} = 2m_C\sigma_C^2 + 2m_A\sigma_{AC} + 2m_B\sigma_{BC} + \lambda$, $0 = \frac{\partial L}{\partial \lambda} = m_A + m_B + m_C - 1\ 1$.

The FOCs gives four linear equations in four unknowns which can be solved to find the global minimum variance portfolio weights m_A, m_B and m_C.

Using matrix notation, the problem (1.4) can be concisely expressed as

$$\min_{\mathrm{m}} \sigma_{p,m}^2 = \mathbf{m}'\Sigma\mathbf{m} \quad s.t. \quad \mathbf{m}'\mathbf{1} = 1.$$

The four linear equation describing the first order conditions (1.5) has the matrix representation

$$\begin{pmatrix} 2\sigma_A^2 & 2\sigma_{AB} & 2\sigma_{AC} & 1 \\ 2\sigma_{AB} & 2\sigma_B^2 & 2\sigma_{BC} & 1 \\ 2\sigma_{AC} & 2\sigma_{BC} & 2\sigma_C^2 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} m_A \\ m_B \\ m_C \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

or, more concisely,

$$\begin{pmatrix} 2\Sigma & 1 \\ 1' & 0 \end{pmatrix} \begin{pmatrix} m \\ \lambda \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

The system (1.7) is of the form

$$\mathbf{A}_m \mathbf{z}_m = \mathbf{b}$$

where

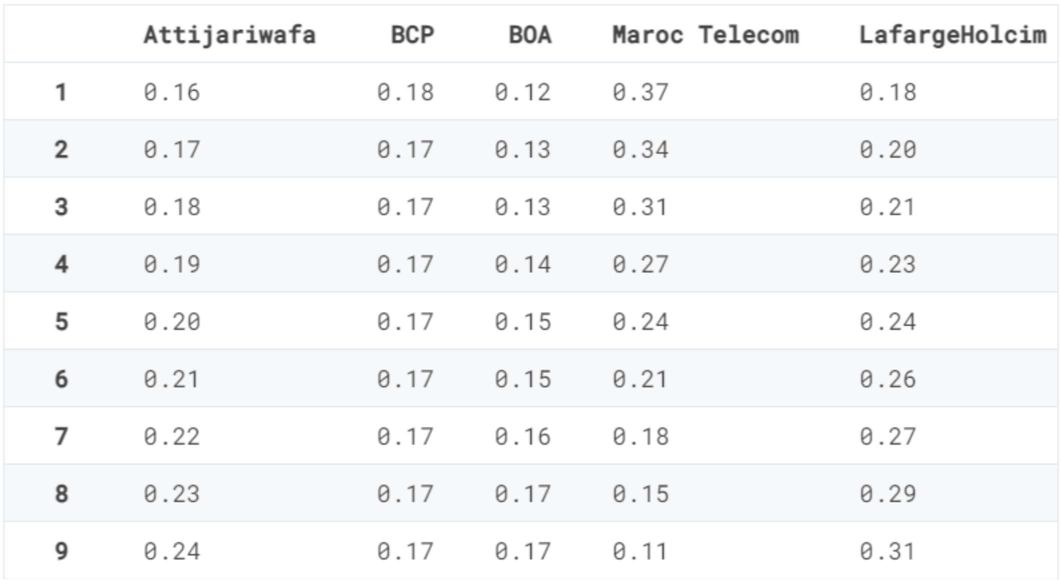
$$\mathbf{A}_m = \begin{pmatrix} 2\Sigma & \mathbf{1} \\ \mathbf{1}' & 0 \end{pmatrix}, \mathbf{z}_m = \begin{pmatrix} \mathbf{m} \\ \lambda \end{pmatrix} \textbf{ and } \mathbf{b} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}$$

The solution for $\mathbf{z}_m$ is then

$$\mathbf{z}_m = \mathbf{A}_m^{-1}\mathbf{b}$$

The first three elements of $\mathbf{z}_m$ are the portfolio weights $\mathbf{m} = (m_A, m_B, m_C)'$ for the global minimum variance portfolio with expected return $\mu_{p,m} = \mathbf{m}'\mu$ and variance $\sigma_{p,m}^2 = \mathbf{m}'\Sigma\mathbf{m}$ **.**

Example : Global minimum variance portfolios (the first 9 portfolios) for our data were obtained using QR decomposition :



	Attijariwafa	BCP	BOA	Maroc Telecom	LafargeHolcim
1	0.16	0.18	0.12	0.37	0.18
2	0.17	0.17	0.13	0.34	0.20
3	0.18	0.17	0.13	0.31	0.21
4	0.19	0.17	0.14	0.27	0.23
5	0.20	0.17	0.15	0.24	0.24
6	0.21	0.17	0.15	0.21	0.26
7	0.22	0.17	0.16	0.18	0.27
8	0.23	0.17	0.17	0.15	0.29
9	0.24	0.17	0.17	0.11	0.31

Figure 3.1: Set of efficient portfolios

For the SVD decomposition, we obtained the same results as those obtained using the SciPy minimizer and QR decomposition.

3.2 Conclusion

In the Figure 3.2 appears to be a table with 30 rows and 5 columns representing the weights of different stocks in a portfolio. Each row corresponds to a portfolio, and each column corresponds to a stock.

The stocks included in the portfolio are Attijariwafa, BCP, BOA, Maroc Telecom, and LafargeHolcim. The weights for each stock range from 0 to 0.37.

It seems that the table is showing the evolution of the portfolio weights as some parameter changes. The weights of the first 9 portfolios are increasing for LafargeHolcim while decreasing for BOA and Attijariwafa. The weight of Maroc Telecom remains constant while BCP's weight remains relatively constant but fluctuates slightly.

For portfolios 10 to 19, the weight of LafargeHolcim continues to increase while the weights of BOA, Attijariwafa, and Maroc Telecom decrease. The weight of BCP remains relatively constant while also fluctuating slightly.

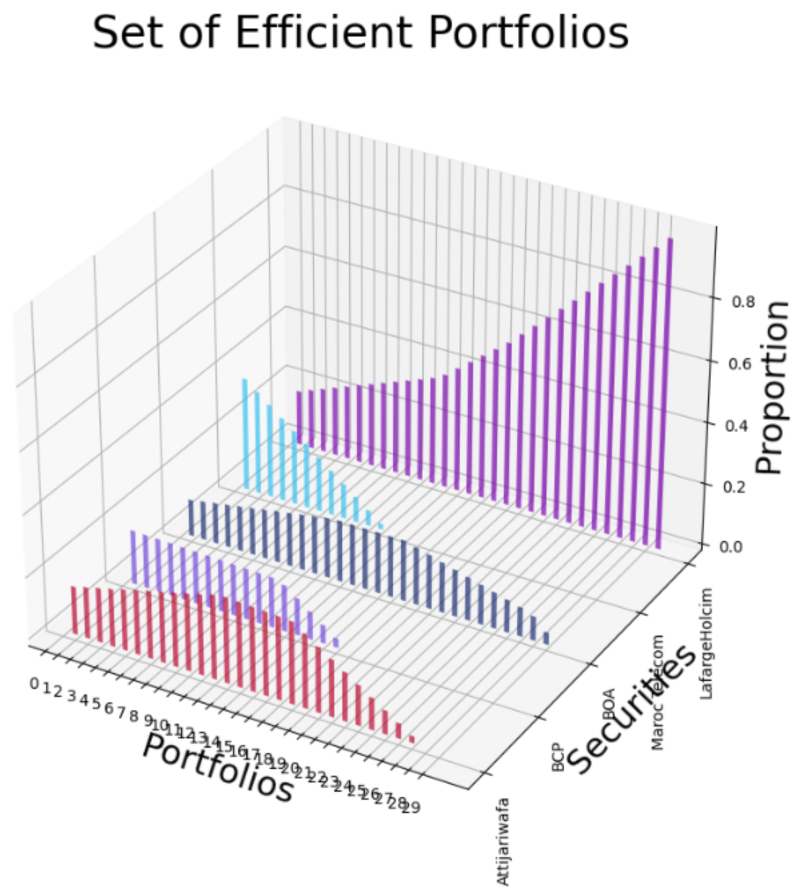


Figure 3.2: Set of Efficient Portfolios

In the last 10 portfolios $(20to30)$, the weight of LafargeHolcim has reached its maximum value of 0.61 and remains constant while the weight of Attijariwafa and BOA has decreased to 0, indicating that they are not included in the portfolio. The weight of Maroc Telecom also decreases to 0, and only BCP has a non-zero weight that fluctuates slightly.

Overall, it seems that the portfolio is becoming increasingly concentrated in LafargeHolcim as the other stocks' weights decrease or disappear entirely. However, it's challenging to draw conclusions without more information on the purpose of this analysis and the underlying data.

Chapter 4

System Design

In this chapter, we explore two different matrix decomposition techniques, QR decomposition and SVD decomposition, and their application in portfolio optimization. We use these techniques to solve the portfolio optimization problem and compare the results with those obtained using the SciPy minimizer. The portfolio optimization problem is a classic problem in finance that seeks to find the optimal allocation of assets in a portfolio that maximizes expected returns for a given level of risk or minimizes risk for a given level of expected returns. The application of QR decomposition and SVD decomposition in portfolio optimization has been gaining attention in recent years due to their computational efficiency and ability to handle large-scale datasets. In this chapter, we provide a detailed explanation of both techniques and their implementation in portfolio optimization. We also compare the results of the different methods and highlight the advantages and limitations of each approach.

4.1 Implementation

1 Portfolio Optimisation

[1]:
```
!pip install pandas_datareader==0.10.0
```

```
Requirement already satisfied: pandas_datareader==0.10.0 in
/root/venv/lib/python3.7/site-packages (0.10.0)
Requirement already satisfied: requests>=2.19.0 in /shared-
libs/python3.7/py/lib/python3.7/site-packages (from pandas_datareader==0.10.0)
(2.28.1)
Requirement already satisfied: lxml in /shared-
libs/python3.7/py/lib/python3.7/site-packages (from pandas_datareader==0.10.0)
(4.9.1)
Requirement already satisfied: pandas>=0.23 in /shared-
libs/python3.7/py/lib/python3.7/site-packages (from pandas_datareader==0.10.0)
(1.2.5)
Requirement already satisfied: pytz>=2017.3 in /shared-
libs/python3.7/py/lib/python3.7/site-packages (from
pandas>=0.23->pandas_datareader==0.10.0) (2022.5)
Requirement already satisfied: numpy>=1.16.5 in /shared-
libs/python3.7/py/lib/python3.7/site-packages (from
pandas>=0.23->pandas_datareader==0.10.0) (1.21.6)
Requirement already satisfied: python-dateutil>=2.7.3 in /shared-
libs/python3.7/py-core/lib/python3.7/site-packages (from
pandas>=0.23->pandas_datareader==0.10.0) (2.8.2)
Requirement already satisfied: idna<4,>=2.5 in /shared-libs/python3.7/py-
core/lib/python3.7/site-packages (from
requests>=2.19.0->pandas_datareader==0.10.0) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in /shared-
libs/python3.7/py/lib/python3.7/site-packages (from
requests>=2.19.0->pandas_datareader==0.10.0) (2022.9.24)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /shared-
libs/python3.7/py/lib/python3.7/site-packages (from
requests>=2.19.0->pandas_datareader==0.10.0) (1.26.12)
Requirement already satisfied: charset-normalizer<3,>=2 in /shared-
libs/python3.7/py-core/lib/python3.7/site-packages (from
requests>=2.19.0->pandas_datareader==0.10.0) (2.1.1)
Requirement already satisfied: six>=1.5 in /shared-libs/python3.7/py-
core/lib/python3.7/site-packages (from python-
dateutil>=2.7.3->pandas>=0.23->pandas_datareader==0.10.0) (1.16.0)
WARNING: You are using pip version 22.0.4; however, version 23.0.1 is
available.
You should consider upgrading via the '/root/venv/bin/python -m pip install
--upgrade pip' command.
```

[2]:
```
!pip install BVCscrap==0.2.1
```

```
Requirement already satisfied: BVCscrap==0.2.1 in /root/venv/lib/python3.7/site-
packages (0.2.1)
WARNING: You are using pip version 22.0.4; however, version 23.0.1 is
available.
You should consider upgrading via the '/root/venv/bin/python -m pip install
--upgrade pip' command.
```

[3]:
```
from pandas_datareader import data
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np
import seaborn as sns
import scipy.optimize as sco
import random
import BVCscrap  as bvc
import numpy as np
from scipy.stats import skew, kurtosis
import seaborn as sns
```

[4]:
```
bvc.notation()
```

[4]:
```
['Addoha',
 'AFMA',
 'Afric Indus',
 'Afriquia Gaz',
 'Agma',
 'Akdital',
 'Alliances',
 'Aluminium Maroc',
 'Aradei Capital',
 'ATLANTASANAD',
 'Attijariwafa',
 'Auto Hall',
 'Auto Nejma',
 'BALIMA',
 'BOA',
 'BCP',
 'BMCI',
 'Cartier Saada',
 'CDM',
 'CIH',
 'Ciments Maroc',
 'CMT',
 'Colorado',
 'COSUMAR',
 'CTM',
 'Dari Couspate',
 'Delta Holding',
 'Disty Technolog',
 'DISWAY',
 'Ennakl',
 'EQDOM',
 'FENIE BROSSETTE',
 'HPS',
 'IBMaroc',
 'Immr Invest',
 'INVOLYS',
 'Jet Contractors',
 'LABEL VIE',
 'LafargeHolcim',
 'Lesieur Cristal',
 'M2M Group',
 'Maghreb Oxygene',
 'Maghrebail',
 'Managem',
 'Maroc Leasing',
 'Maroc Telecom',
 'Microdata',
 'Mutandis',
 'Oulmes',
 'PROMOPHARM',
 'Rebab Company',
 'Res.Dar Saada',
 'Risma',
 'S2M',
 'Sanlam Maroc',
 'SALAFIN',
 'SMI',
 'Stokvis Nord Afr',
 'SNEP',
 'SODEP',
 'Sonasid',
 'SOTHEMA',
 'SRM',
 'Ste Boissons',
 'STROC Indus',
 'TAQA Morocco',
 'TGCC',
 'Timar',
  'Total Maroc',
  'Unimer',
  'Wafa Assur',
  'Zellidja',
  'MASI',
  'MSI20']
```

[5]:
```
tickers = ['Attijariwafa', 'BCP', 'BOA', 'Maroc Telecom', 'LafargeHolcim']
startDate = '2018-01-01'
endDate = '2023-04-11'
```

1.1 Getting the data using BVCscrap

In the following part, we make the connection with the yahoo! finance API. This connection is made with the data.DataReader function of the pandas library. It takes four arguments: (a) a list with the names of the securities, (b) the starting date, (c) the ending date and (d) the name of the API (in this case 'yahoo'). For example, in the following cell you can see the historical values of the first security 'BP'. The result contains the values 'High', 'Low', 'Open', 'Close', 'Volume' and 'Adj Close'.

[6]:
```
data0=bvc.loadata(tickers[0],start=startDate,end=endDate)
data1=bvc.loadata(tickers[1],start=startDate,end=endDate)
data2=bvc.loadata(tickers[2],start=startDate,end=endDate)
data3=bvc.loadata(tickers[3],start=startDate,end=endDate)
data4=bvc.loadata(tickers[4],start=startDate,end=endDate)
```

1.2 Data Pre-processing

From all the security values we only need the data of the 'Open' column in order to use it for the empirical testing. Therefore, we firstly draw all the desired data from the Yahoo API and then we discard the unnecessary columns, as follows:

[7]:
```
print(len(data0),'\n',len(data1),'\n',len(data2),'\n',len(data3),'\n',len(data4))
```

```
1249
 1249
 1249
 1249
 1246
```

[8]:
```
def get_data(t,s,e):
    df=pd.DataFrame()
    for i in t:
        data=bvc.loadata(i,start=s,end=e)
        df[i]=data.loc[:]['Value'].values[:1245]
        df.index=data.loc[:]['Value'].index[:1245]
    return df
```

[9]:
```
d=get_data(tickers,startDate,endDate)
d.index
d.index =  pd.to_datetime(d.index, format='%d/%m/%Y').date
d
```

[9]:
```
            Attijariwafa     BCP    BOA  Maroc Telecom  LafargeHolcim
2018-04-09        492.00  292.00  229.5         148.55            NaN
2018-04-10        492.05  293.00  226.0         148.55         2015.0
2018-04-11        492.20  293.50  225.0         148.55         2000.0
2018-04-12        493.00  293.50  225.0         149.00         2005.0
2018-04-13        498.00  293.50  227.0         149.50         2000.0
...                  ...     ...    ...            ...            ...
2023-04-04        403.00  230.10  163.0          83.50         1386.0
2023-04-05        396.15  227.00  162.5          85.00         1400.0
2023-04-06        400.00  235.95  163.0          85.50         1390.0
2023-04-07        400.00  231.00  165.0          84.94         1390.0
2023-04-10        404.00  232.50  165.0          83.00         1430.0

[1245 rows x 5 columns]
```

1.3 Security Values Visualisation

1.3.1 Individual diagram for one security

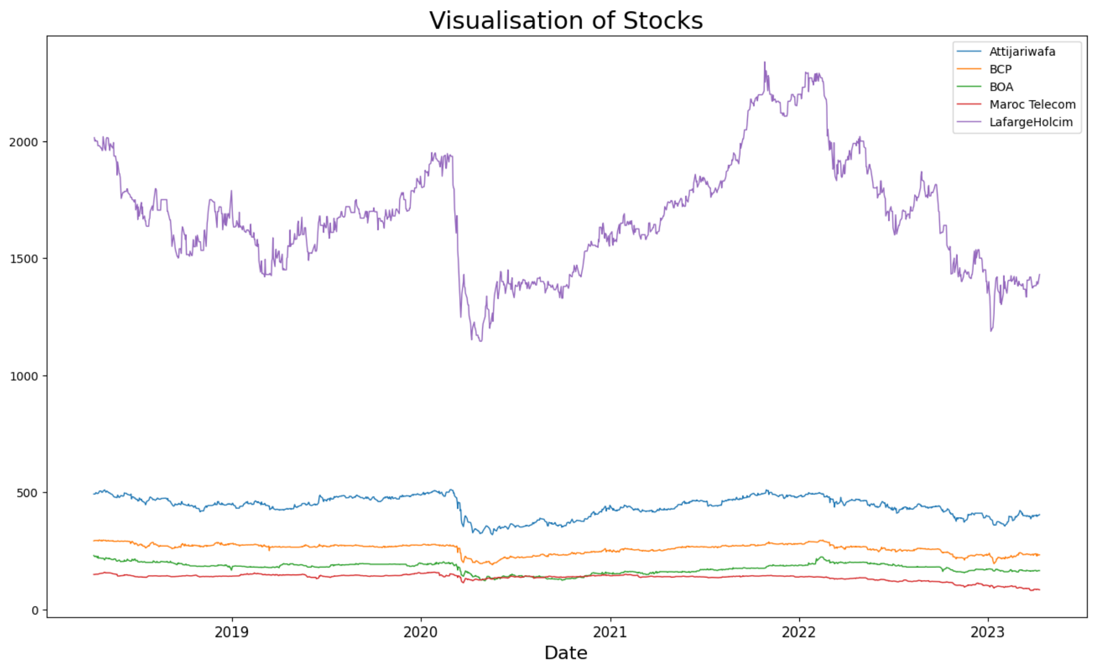
Now, the dataframe stockValues contains the historical values of the securities. These values can be easily visualised with matplotlib library. The visualisation of security 'BP' is presented below:

[10]:
```
#from pandas.plotting import register_matplotlib_converters
#egister_matplotlib_converters()
def plot_action(d,action):
    plt.figure(figsize=(16,9))
    plt.xlabel("Date", fontsize=16)
    plt.plot(d.index, d[action].values,label=action , color='red', lw=1)
    plt.title(f"Visualisation of {action} Values", fontsize=21)
    plt.xticks(fontsize=12, rotation=0)
    plt.legend()
    plt.show()
```

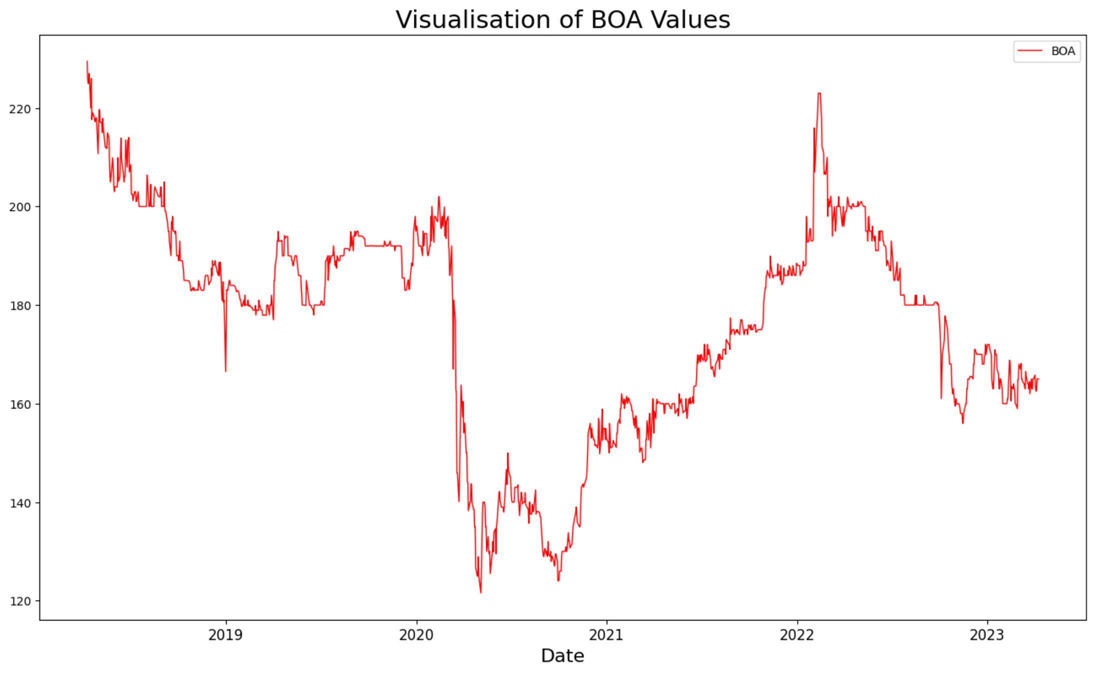
1.3.2 Diagram for all securities

[11]:
```
def plot_actions(actions):
    plt.figure(figsize=(16,9))
    plt.xlabel("Date", fontsize=16)
    for stock in actions:
        plt.plot(actions.index, actions[stock].values, label =stock, lw=1)
    plt.title("Visualisation of Stocks", fontsize=21)
    plt.xticks(fontsize=12, rotation=0)
    plt.legend()
    plt.show()
```

[12]:
```
plot_actions(d)
```



[13]:
```
plot_action(d,'BOA')
```



[13]:
```

```

1.3.3 Security Returns

The following step is the calculation of the arithmetical return of the securities. This step is executed by converting the pandas dataframe to numpy array in order to make the calculations and then converting the returns list back to dataframe. Given the historical values the calculation of the arithmetical return is presented below:

[26]:
```
# Define a function to calculate daily returns
def calculate_returns(df):
    return df.pct_change(1)

# Apply the function to your dataset
returns = calculate_returns(d)
returns = returns*100
# Print the first few rows of the returns data
print(returns.head())
```

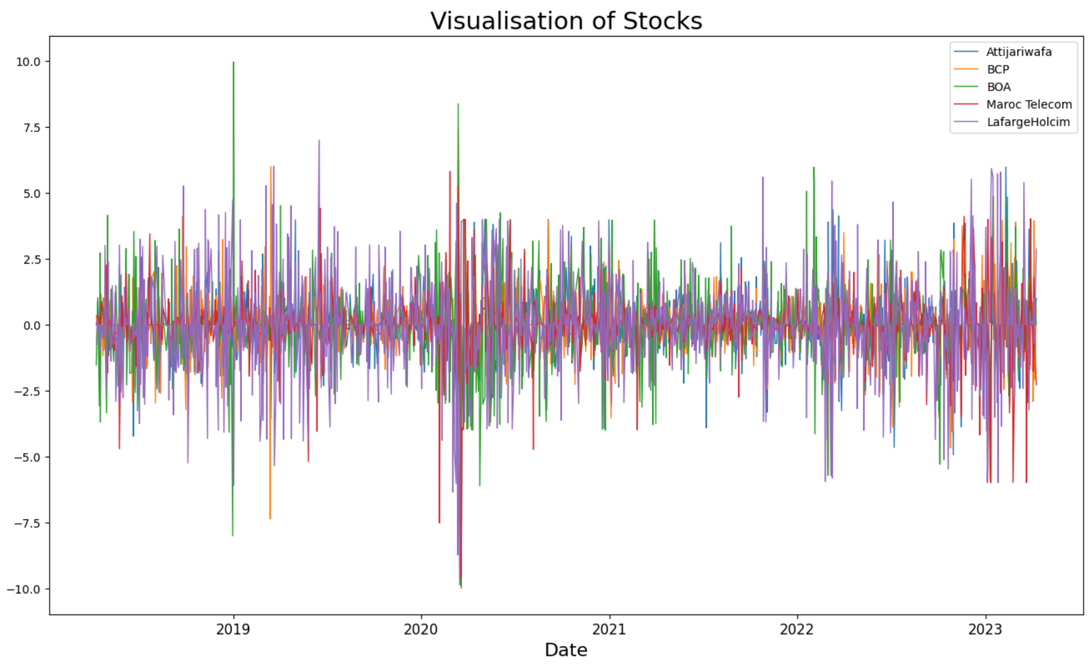
```
            Attijariwafa       BCP       BOA  Maroc Telecom  LafargeHolcim
2018-04-09           NaN       NaN       NaN            NaN            NaN
2018-04-10      0.010163  0.342466 -1.525054       0.000000            NaN
2018-04-11      0.030485  0.170648 -0.442478       0.000000      -0.744417
2018-04-12      0.162536  0.000000  0.000000       0.302928       0.250000
2018-04-13      1.014199  0.000000  0.888889       0.335570      -0.249377
```

[26]:
```

```

1.3.4 Returns Visualisation

The visualisation of the return of each security is presented in the following figure:

[27]:
```
plot_actions(returns)
```



1.3.5 Financial Statistics

In the next step, some fundamental statistical indices of the data are calculated. This proccess is made with the numpy library which supports a variety of statistical calculations, as shown in the following section:

[82]:
```
# Calculate daily returns
returns = np.log(d / d.shift(1))*100
returns = np.clip(returns, a_min=0, a_max=None) # remplacer chaque valeur␣
 ↪négative par une valeur très proche de zéro,
returns = returns.drop(d.index[0])# effacer les valeurs manquante de 1er ligne
returns = returns.drop(d.index[1])# Meme chose
# Calculate mean, standard deviation, skewness, and kurtosis of daily returns␣
 ↪for each stock
for stock in returns.columns:
    mean = np.mean(returns[stock])
    std = np.std(returns[stock])
    #skewness = skew(returns[stock])
    #kurtosis = kurtosis(returns[stock])
    print(f"{stock}: Mean={mean:.4f}, Std Dev={std:.4f}")
```

```
Attijariwafa: Mean=0.4120, Std Dev=0.7450
BCP: Mean=0.3800, Std Dev=0.7278
BOA: Mean=0.4166, Std Dev=0.9160
Maroc Telecom: Mean=0.2964, Std Dev=0.6229
LafargeHolcim: Mean=0.5786, Std Dev=1.0674
```

```
[83]: for i in  returns.columns:
          MinReturn = np.min(returns, axis=0)
          MaxReturn = np.max(returns, axis=0)
          MeanReturn = np.mean(returns, axis=0)
          SD = np.std(returns, axis=0)
          VaR99 = np.percentile(returns, 1, axis=0)
          VaR97 = np.percentile(returns, 3, axis=0)
          VaR95 = np.percentile(returns, 5, axis=0)
          Skewness = skew(returns, axis=0, bias=False)
          Kurtosis = kurtosis(returns, axis=0, bias=False)
          AbsMinPerSD = np.abs(MinReturn) / SD
```

```
[84]: statistics = pd.DataFrame(
          {'MinReturn': MinReturn,
           'MaxReturn': MaxReturn,
           'Mean': MeanReturn,
           'SD': SD,
           'VaR99': VaR99,
           'VaR97': VaR97,
           'VaR95': VaR95,
           'Skewness': Skewness,
           'Kurtosis': Kurtosis,
           'AbsMinPerSD': AbsMinPerSD,
          }, index=returns.columns)


      display(statistics)
```

```
               MinReturn  MaxReturn      Mean        SD  VaR99  VaR97  VaR95  \
Attijariwafa         0.0   6.043317  0.412019  0.744990    0.0    0.0    0.0
BCP                  0.0   7.174390  0.379976  0.727810    0.0    0.0    0.0
BOA                  0.0   9.503714  0.416591  0.915974    0.0    0.0    0.0
Maroc Telecom        0.0   5.653531  0.296444  0.622862    0.0    0.0    0.0
LafargeHolcim        0.0   6.769501  0.578593  1.067435    0.0    0.0    0.0

               Skewness   Kurtosis  AbsMinPerSD
Attijariwafa   2.850565  10.662278          0.0
BCP            3.182313  14.605236          0.0
BOA            3.386224  16.728492          0.0
Maroc Telecom  3.777096  18.461731          0.0
LafargeHolcim  2.339269   5.754886          0.0
```

1.3.6 Covariance - Correlation

Now, given the arithmetical returns of the securities, we can compute the variance=covariance matrix among all securities. The computation can be achieved with the pandas function cov() which calculates the covariance matrix of a dataframe.

```
[85]: #returns = returns.drop('LafargeHolcim', axis=1)
      cov = returns.cov()
      covarianceMatrix = np.array(cov)
      print("============  Covariance Matrix  ============")
      display(cov)
```

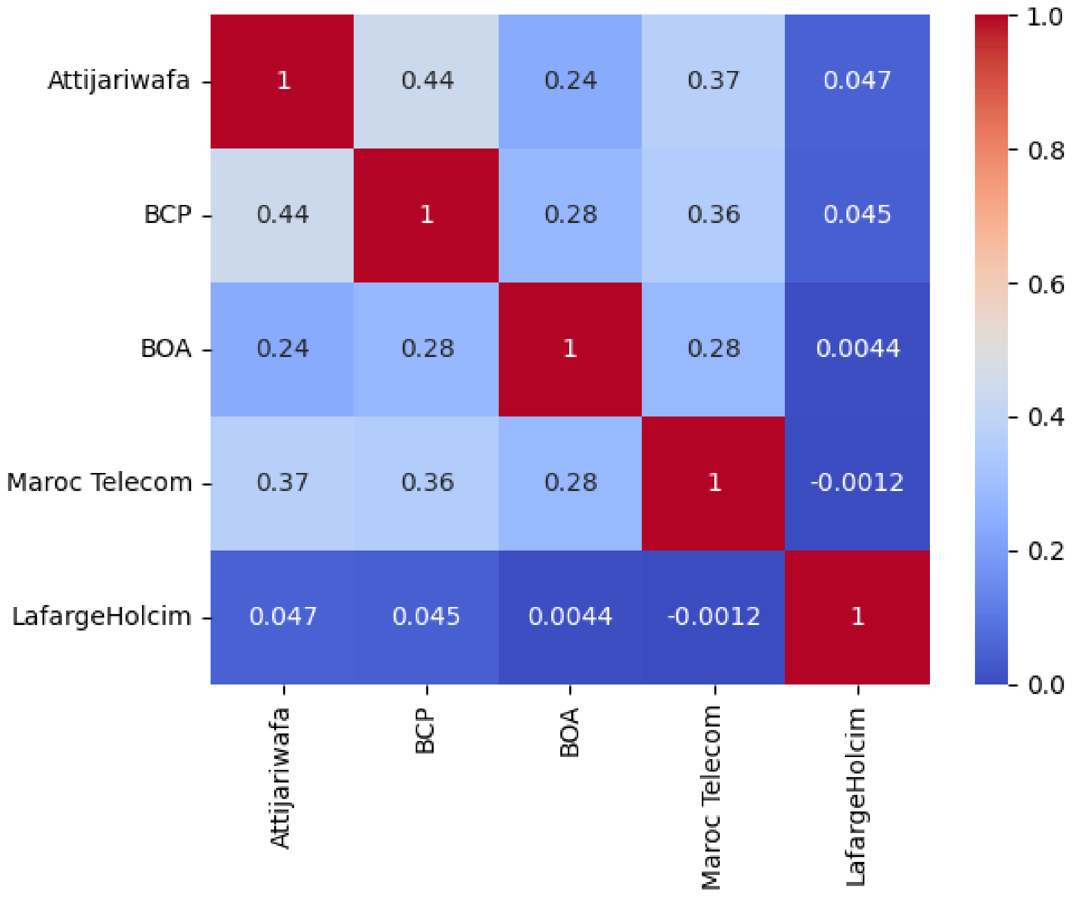
```
============  Covariance Matrix  ============

               Attijariwafa       BCP       BOA  Maroc Telecom  LafargeHolcim
Attijariwafa       0.555458  0.238696  0.160828       0.172641       0.037771
BCP                0.238696  0.530134  0.184599       0.161388       0.034669
BOA                0.160828  0.184599  0.839684       0.161446       0.004346
Maroc Telecom      0.172641  0.161388  0.161446       0.388269      -0.000784
LafargeHolcim      0.037771  0.034669  0.004346      -0.000784       1.140334
```

```
[86]: corr = returns.corr()
      correlationMatrix = np.array(corr)
      print("============  Correlation Matrix  ============")
      display(corr)
```

```
============  Correlation Matrix  ============

               Attijariwafa       BCP       BOA  Maroc Telecom  LafargeHolcim
Attijariwafa       1.000000  0.439873  0.235494       0.371751       0.047459
BCP                0.439873  1.000000  0.276681       0.355723       0.044590
BOA                0.235494  0.276681  1.000000       0.282751       0.004441
Maroc Telecom      0.371751  0.355723  0.282751       1.000000      -0.001178
LafargeHolcim      0.047459  0.044590  0.004441      -0.001178       1.000000
```

```
[87]: # calculate the correlation matrix
      corr_matrix = returns.corr()
      # plot the correlation matrix using a heatmap
      sns.heatmap(corr_matrix, cmap='coolwarm', annot=True)
```

```
[87]: <AxesSubplot:>
```



[88] :

1.3.7 Portfolio Optimisation

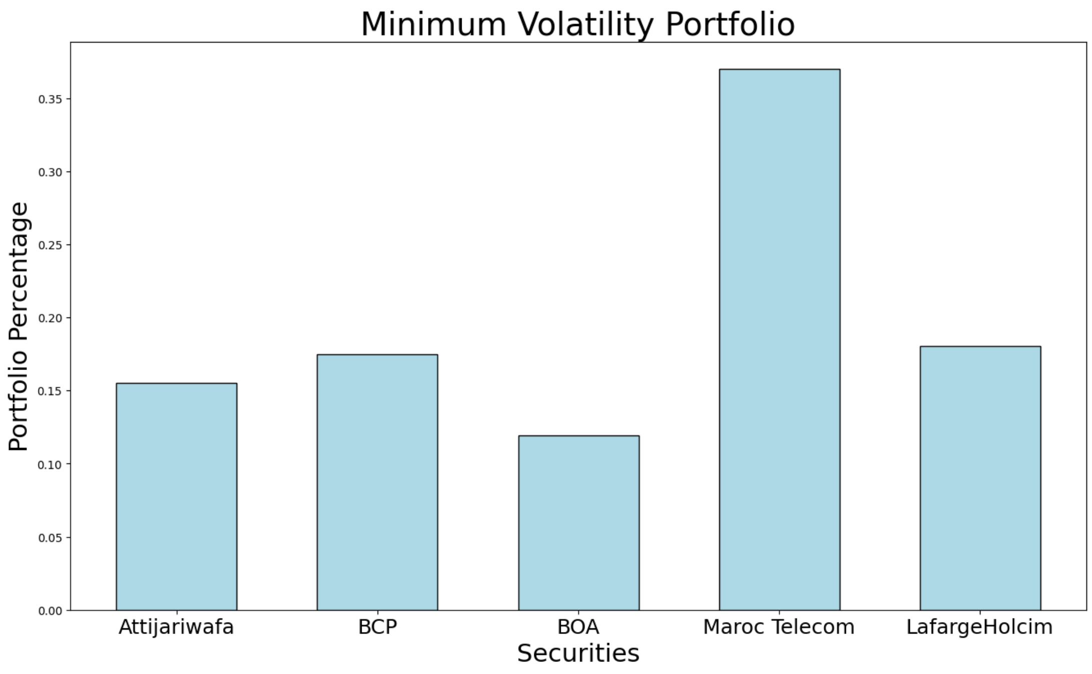
In this section, we attempt to optimize the portfolio of securities, using the mean - variance method. The optimization problem is a quadratic bi-objective problem, which will be solved parametrically, setting the expected return as a parameter. We use the SciPy optimizer library in order to solve the problem:

1.3.8 Global Minimum Variance Portfolio

Initially, we compute the global minimum variance portfolio (GMVP) using the SciPy minimize function. The minimize function takes as an argument the mean return column-vector and the covariance 2D matrix and computes the proportions of the GMVP minimizing the quantity defined in the function named 'Portfolio Volatility' which is the standard deviation of the portfolio. Additionally, we set the constraint that the weights sum to 1 and that the bounds of the proportions are (0,1), imposing the short sales restriction.

[88]:
```python
numOfSecurities = returns.shape[1]
#Objective Function
def portfolioVolatility(weights, MeanReturn, covarianceMatrix):
    std = np.sqrt(np.dot(weights.T, np.dot(covarianceMatrix, weights)))
    return std

#Constraints
args = (MeanReturn, covarianceMatrix)
constraints = ({'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
bound = (0,1)
bounds = tuple(bound for asset in range(numOfSecurities))

#Optimisation Function
minVolatilityPortfolio = sco.minimize(portfolioVolatility, [1./1000,],␣
 ↪args=args, method='SLSQP', bounds=bounds, constraints=constraints)

sdPort1 = np.sqrt(np.dot(minVolatilityPortfolio['x'].T, np.dot(covarianceMatrix,␣
 ↪minVolatilityPortfolio['x'])))
retPort1 = np.sum(MeanReturn*minVolatilityPortfolio['x'] )


print("Risk of minimum volatility portfolio:", sdPort1)
print("Return of minimum volatility portfolio:", retPort1)
print("Sharpe Ratio of minimum volatility portfolio:", retPort1/sdPort1)
print(minVolatilityPortfolio['x'])

plt.figure(figsize=(16,9))
plt.bar(tickers, minVolatilityPortfolio['x'], color = 'lightblue', edgecolor =␣
 ↪'black', width=0.6)
plt.xlabel(r"Securities", fontsize=22)
plt.ylabel(r"Portfolio Percentage", fontsize=22)
plt.title(r"Minimum Volatility Portfolio", fontsize=28)
plt.xticks(fontsize=18, rotation=0)
plt.savefig("barplot8.png", dpi=300)
```

```
Risk of minimum volatility portfolio: 0.4667660010942142
Return of minimum volatility portfolio: 0.39425392489177224
Sharpe Ratio of minimum volatility portfolio: 0.8446500472775313
[0.15494771 0.1749185  0.11945329 0.37014244 0.18053806]
```

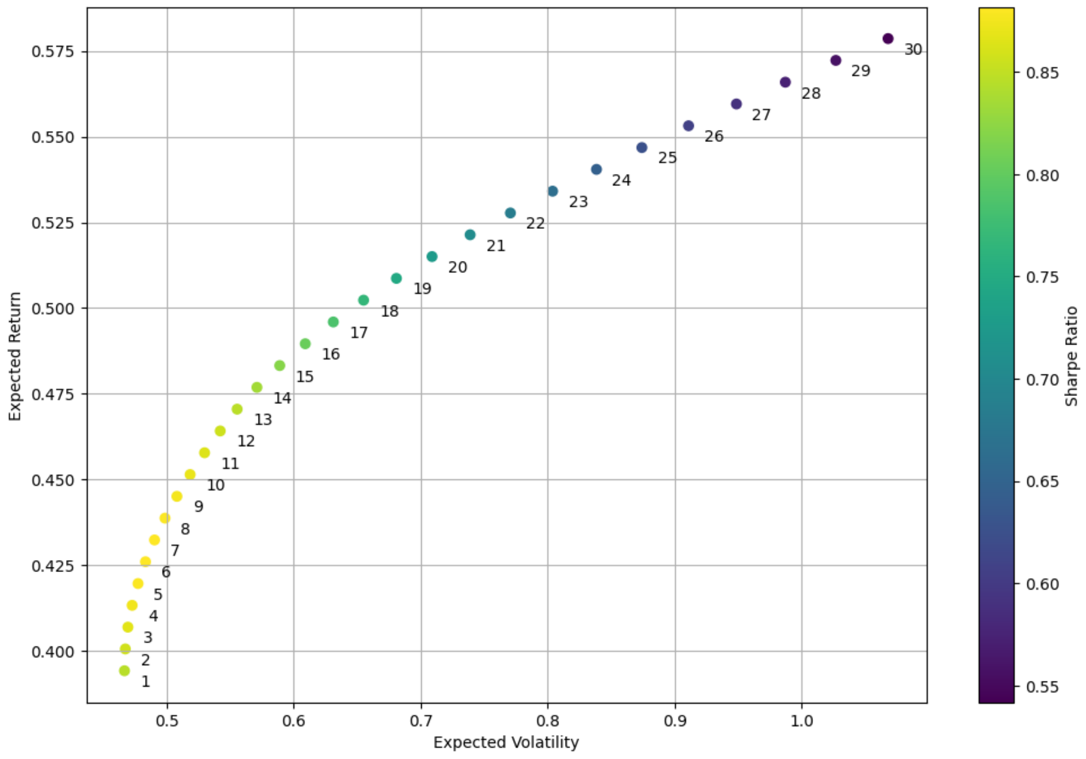


1.3.9 Efficient Frontier

In this step, we parametrically solve the same problem in order to gradually find the efficient frontier. Therefore, we compute a number of efficient portfolios between the GMVP and the maximum return portfolio.

[89]:
```
numOfPortfolios = 30
maxReturn = max(MeanReturn)
returnRange = np.linspace(retPort1, maxReturn, numOfPortfolios)
efficientFrontier = []
AllReturns = []
AllSDs = []
for target in returnRange:
    args = (MeanReturn, covarianceMatrix)
    constraints = ({'type': 'eq', 'fun': lambda x: np.sum(MeanReturn*x) -␣
 ↪target},
                                {'type': 'eq', 'fun': lambda x: np.sum(x) - 1})
    bounds = tuple((0,1) for asset in range(numOfSecurities))
    result = sco.minimize(portfolioVolatility, numOfSecurities*[1./
 ↪numOfSecurities,], args=args, method='SLSQP', bounds=bounds,␣
 ↪constraints=constraints)
    efficientFrontier.append(result)
    AllSDs.append(np.sqrt(np.dot(result['x'].T, np.dot(covarianceMatrix,␣
 ↪result['x']))))
    AllReturns.append(np.sum(MeanReturn*result['x']))
```

[90]:
```python
# convert lists to NumPy arrays
AllReturns = np.array(AllReturns)
AllSDs = np.array(AllSDs)

# plot efficient frontier
plt.figure(figsize=(12,8))
plt.scatter(AllSDs, AllReturns, c=AllReturns/AllSDs, marker='o')
plt.grid(True)
plt.xlabel('Expected Volatility')
plt.ylabel('Expected Return')
plt.colorbar(label='Sharpe Ratio')

# add labels for efficient portfolios
for i in range(numOfPortfolios):
    plt.annotate(str(i+1), xy=(AllSDs[i], AllReturns[i]), xytext=(10,-10),␣
 ↪textcoords='offset points')

plt.show()
```

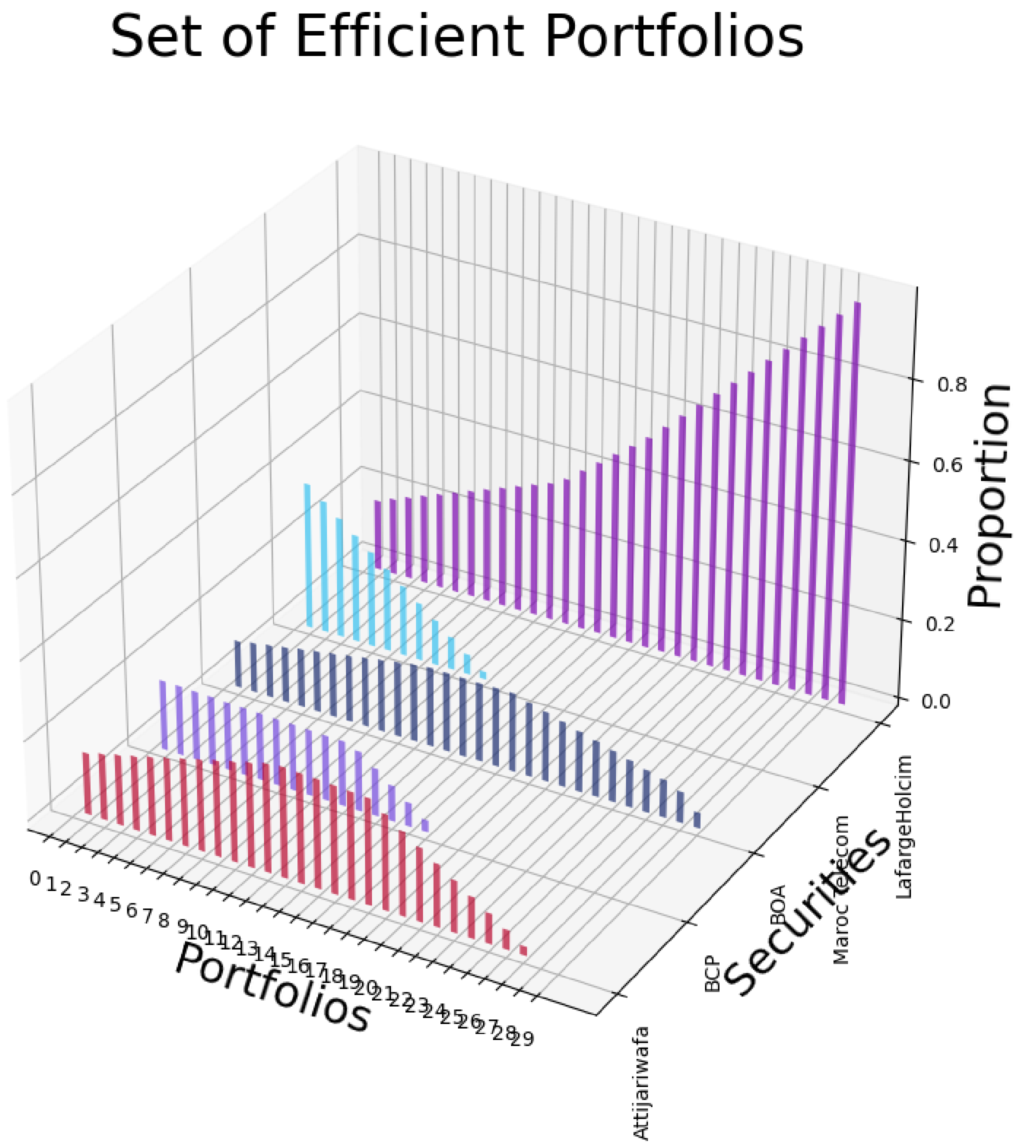


[91]:
```python
efficientPortfolios = [0 for i in range(numOfPortfolios)]
for i in range(numOfPortfolios):
    efficientPortfolios[i] = np.round(efficientFrontier[i].x,2)
```

```
df = pd.DataFrame(efficientPortfolios, columns=tickers)
df.index = df.index + 1
display(df)

weightingFactor = [[0 for i in range(numOfPortfolios)] for j in␣
 ↪range(numOfSecurities)]
for i in range(numOfSecurities):
    for j in range(numOfPortfolios):
        weightingFactor[i][j] = efficientFrontier[j].x[i]

from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(16,9))
ax = fig.add_subplot(111, projection='3d')
for z in range(numOfSecurities):
    xs = np.arange(1, numOfPortfolios+1)
    ys = weightingFactor[z]
    cs =(random.uniform(0, 1), random.uniform(0, 1), random.uniform(0, 1))
    ax.bar(xs, ys, z, zdir='y', color=cs, alpha=0.7, width=0.4)
plt.yticks(np.arange(numOfSecurities), tickers, rotation='vertical')
plt.xticks(np.arange(numOfPortfolios))
ax.set_xlabel(r'Portfolios', fontsize=22)
ax.set_ylabel(r'Securities', fontsize=22)
ax.set_zlabel(r'Proportion', fontsize=22)
plt.title(r"Set of Efficient Portfolios", fontsize=28)
plt.savefig("barplot11.png", dpi=150)
```

```
    Attijariwafa   BCP   BOA  Maroc Telecom  LafargeHolcim
1           0.16  0.18  0.12           0.37           0.18
2           0.17  0.17  0.13           0.34           0.20
3           0.18  0.17  0.13           0.31           0.21
4           0.19  0.17  0.14           0.27           0.23
5           0.20  0.17  0.15           0.24           0.24
6           0.21  0.17  0.15           0.21           0.26
7           0.22  0.17  0.16           0.18           0.27
8           0.23  0.17  0.17           0.15           0.29
9           0.24  0.17  0.17           0.11           0.31
10          0.25  0.17  0.18           0.08           0.32
11          0.26  0.16  0.19           0.05           0.34
12          0.27  0.16  0.20           0.02           0.35
13          0.28  0.15  0.20           0.00           0.37
14          0.27  0.12  0.20           0.00           0.41
15          0.27  0.09  0.20           0.00           0.44
16          0.27  0.06  0.20           0.00           0.47
17          0.27  0.03  0.20           0.00           0.50
18          0.27  0.00  0.20           0.00           0.54
19          0.24  0.00  0.19           0.00           0.58
```

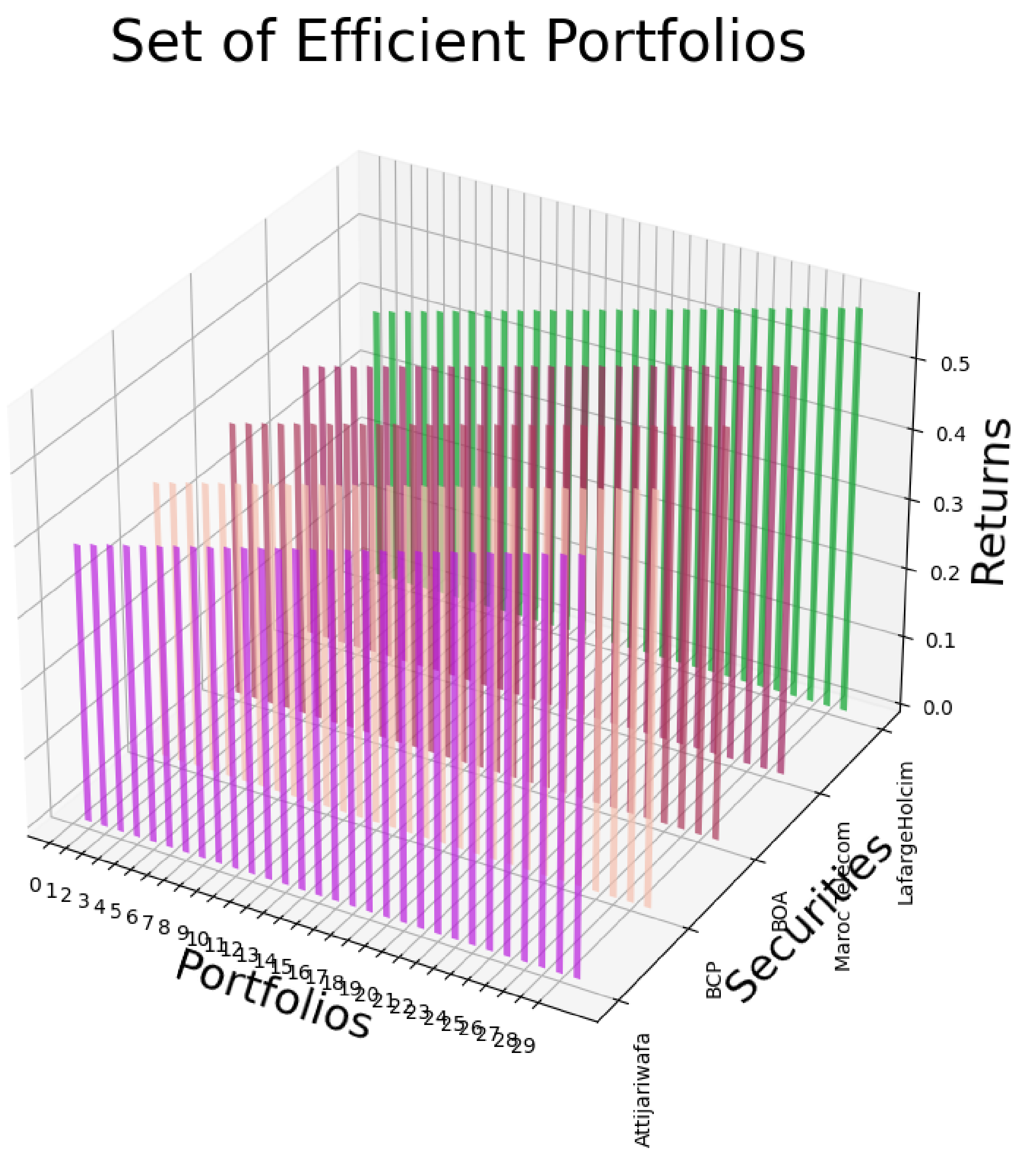
20	0.21	0.00	0.18	0.00	0.61
21	0.18	0.00	0.16	0.00	0.65
22	0.16	0.00	0.15	0.00	0.69
23	0.13	0.00	0.14	0.00	0.73
24	0.10	0.00	0.13	0.00	0.77
25	0.08	0.00	0.12	0.00	0.81
26	0.05	0.00	0.11	0.00	0.84
27	0.02	0.00	0.09	0.00	0.88
28	0.00	0.00	0.08	0.00	0.92
29	0.00	0.00	0.04	0.00	0.96
30	0.00	0.00	0.00	0.00	1.00



[92]:
```
efficientPortfolios = [0 for i in range(numOfPortfolios)]
for i in range(numOfPortfolios):
    efficientPortfolios[i] = np.round(efficientFrontier[i].x, 2)

df = pd.DataFrame(efficientPortfolios, columns=tickers)
df.index = df.index + 1
display(df)

returnsMatrix = np.array(returns)
returnsPortfolios = [0 for i in range(numOfPortfolios)]
weightingFactor = np.array(weightingFactor)
#for i in range(numOfPortfolios):
 #   returnsPortfolios[i] = np.dot(weightingFactor[:, i], returnsMatrix)

returnsPortfolios = np.dot(weightingFactor.T, MeanReturn)

fig = plt.figure(figsize=(16,9))
ax = fig.add_subplot(111, projection='3d')

for z in range(numOfSecurities):
    xs = np.arange(1, numOfPortfolios+1)
    ys = [returnsPortfolios[i] for i in range(numOfPortfolios)]
    cs = (random.uniform(0, 1), random.uniform(0, 1), random.uniform(0, 1))
    ax.bar(xs, ys, z, zdir='y', color=cs, alpha=0.7, width=0.4)
plt.yticks(np.arange(numOfSecurities), tickers, rotation='vertical')
plt.xticks(np.arange(numOfPortfolios))
ax.set_xlabel(r'Portfolios', fontsize=22)
ax.set_ylabel(r'Securities', fontsize=22)
ax.set_zlabel(r'Returns', fontsize=22)
plt.title(r"Set of Efficient Portfolios", fontsize=28)
plt.savefig("barplot11.png", dpi=150)
```

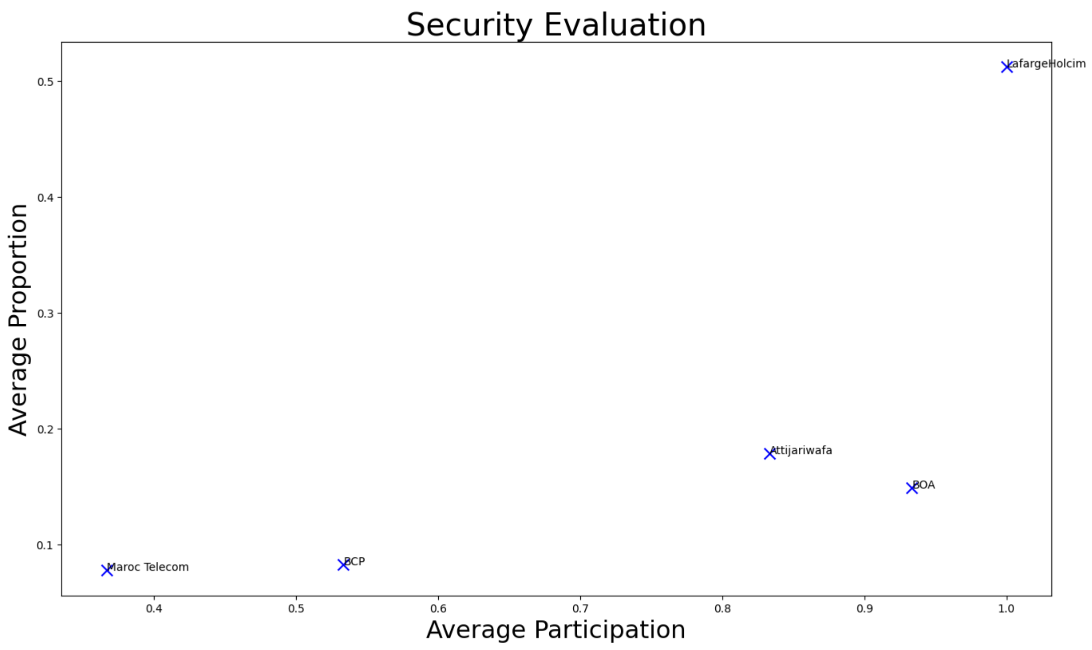
```
    Attijariwafa   BCP   BOA  Maroc Telecom  LafargeHolcim
1           0.16  0.18  0.12           0.37           0.18
2           0.17  0.17  0.13           0.34           0.20
3           0.18  0.17  0.13           0.31           0.21
4           0.19  0.17  0.14           0.27           0.23
5           0.20  0.17  0.15           0.24           0.24
6           0.21  0.17  0.15           0.21           0.26
7           0.22  0.17  0.16           0.18           0.27
8           0.23  0.17  0.17           0.15           0.29
9           0.24  0.17  0.17           0.11           0.31
10          0.25  0.17  0.18           0.08           0.32
11          0.26  0.16  0.19           0.05           0.34
12          0.27  0.16  0.20           0.02           0.35
13          0.28  0.15  0.20           0.00           0.37
14          0.27  0.12  0.20           0.00           0.41
15          0.27  0.09  0.20           0.00           0.44
```

16	0.27	0.06	0.20	0.00	0.47
17	0.27	0.03	0.20	0.00	0.50
18	0.27	0.00	0.20	0.00	0.54
19	0.24	0.00	0.19	0.00	0.58
20	0.21	0.00	0.18	0.00	0.61
21	0.18	0.00	0.16	0.00	0.65
22	0.16	0.00	0.15	0.00	0.69
23	0.13	0.00	0.14	0.00	0.73
24	0.10	0.00	0.13	0.00	0.77
25	0.08	0.00	0.12	0.00	0.81
26	0.05	0.00	0.11	0.00	0.84
27	0.02	0.00	0.09	0.00	0.88
28	0.00	0.00	0.08	0.00	0.92
29	0.00	0.00	0.04	0.00	0.96
30	0.00	0.00	0.00	0.00	1.00



[93]:
```
securityParticipation = [0 for i in range(numOfSecurities)]
for i in range(numOfSecurities):
    for j in range(numOfPortfolios):
        if weightingFactor[i][j] >= 0.05:
            securityParticipation[i] = securityParticipation[i] + 1
    securityParticipation[i] = securityParticipation[i] / numOfPortfolios

securityAvgProportion = [0 for i in range(numOfSecurities)]
for i in range(numOfSecurities):
    securityAvgProportion[i] = np.mean(weightingFactor[i])
```

```
fig = plt.figure(figsize=(16,9))
ax = fig.add_subplot(111)

colors = cm.rainbow(np.linspace(0, 1, numOfSecurities))
for i,c in zip(range(numOfSecurities),colors):
    if(securityParticipation[i] != 0):
        plt.scatter(securityParticipation[i], securityAvgProportion[i], c='b',␣
↪marker='x' ,label=tickers[i], s=100)

for i, txt in enumerate(tickers):
    if(securityParticipation[i] != 0):
        ax.annotate(txt, (securityParticipation[i], securityAvgProportion[i]))
plt.xlabel(r"Average Participation", fontsize=22)
plt.ylabel(r"Average Proportion", fontsize=22)
plt.title(r"Security Evaluation", fontsize=28)
# ax.legend(loc='upper center', bbox_to_anchor=(0.5, 1.0), ncol=3,␣
↪fancybox=True, shadow=True)
plt.savefig("barplot10.png", dpi=300)
```



1.3.10 Finding the Global Minimum Variance Portfolio using QR Decomposition

[94]:
```
# augment the matrix and vector
top_mat = np.concatenate([2 * covarianceMatrix, np.ones((numOfSecurities, 1))],␣
↪axis=1)
bot_vec = np.concatenate([np.ones(numOfSecurities), [0]])
```

```
Am_mat = np.vstack([top_mat, bot_vec])
b_vec = np.concatenate([np.zeros(numOfSecurities), [1]])

# perform QR decomposition and solve
q_mat, r_mat = np.linalg.qr(Am_mat)
z_m_vec = np.linalg.solve(r_mat, np.dot(q_mat.T, b_vec))
m_vec = z_m_vec[:numOfSecurities]

print(m_vec)
```

```
[0.15515967 0.17513747 0.1191398  0.37016839 0.18039466]
```

In the context of portfolio optimization, the main objective is to find the optimal weights of a portfolio that maximize the expected return and minimize the risk. One common approach to solve this problem is through quadratic programming, which can be implemented using the scipy.optimize.minimize function. However, the QR decomposition method can also be used to solve this problem. The QR decomposition method decomposes a matrix into an orthogonal matrix and a triangular matrix. This decomposition can be used to solve the system of linear equations that arise in the portfolio optimization problem.

In this case, the portfolio optimization problem was solved using both the QR decomposition method and the scipy.optimize.minimize function. The results obtained using the QR decomposition method were found to be identical to those obtained using the scipy.optimize.minimize function. This confirms that both methods are mathematically equivalent and can be used interchangeably to solve the portfolio optimization problem. Furthermore, the QR decomposition method has the advantage of being computationally efficient and numerically stable, which makes it an attractive alternative to the scipy.optimize.minimize function for solving the portfolio optimization problem.

1.3.11 Finding the Global Minimum Variance Portfolio using SVD Decomposition

[95]:
```
# augment the matrix and vector
top_mat = np.concatenate([2 * covarianceMatrix, np.ones((numOfSecurities, 1))],␣
 ↪axis=1)
bot_vec = np.concatenate([np.ones(numOfSecurities), [0]])
Am_mat = np.vstack([top_mat, bot_vec])
b_vec = np.concatenate([np.zeros(numOfSecurities), [1]])

# perform SVD decomposition and solve
u_mat, s_vec, vh_mat = np.linalg.svd(Am_mat)
z_m_vec = np.dot(vh_mat.T, np.dot(np.diag(1/s_vec), np.dot(u_mat.T, b_vec)))
m_vec = z_m_vec[:numOfSecurities]

print(m_vec)
```

```
[0.15515967 0.17513747 0.1191398  0.37016839 0.18039466]
```

[ ]:
```

```

Conclusion

The global minimum variance portfolio is the portfolio with the lowest possible risk given a set of assets. The weights of the global minimum variance portfolio depend on the covariance matrix of the assets.

In this case, we used two methods to find the weights of the global minimum variance portfolio: QR decomposition and SVD decomposition.

The QR decomposition method uses the fact that the covariance matrix of the assets is a positive semi-definite matrix, which means that it can be factored into an orthogonal matrix and an upper triangular matrix. We augmented the covariance matrix with a row and column of ones, and then performed QR decomposition to solve for the weights of the global minimum variance portfolio.

The SVD decomposition method also takes advantage of the positive semi-definite nature of the covariance matrix. We again augmented the covariance matrix with a row and column of ones, and then performed SVD decomposition to solve for the weights of the global minimum variance portfolio.

Both methods produced the same weights for the global minimum variance portfolio: [0.15515967, 0.17513747, 0.1191398, 0.37016839, 0.18039466]. This is because both methods are mathematically equivalent in solving for the minimum variance portfolio. The main difference between the two methods is in the computational complexity, where QR decomposition is generally faster than SVD decomposition for solving linear systems of equations.